MASTER EN SCIENCES INFORMATIQUES

Implémentation d'une nouvelle méthode d'aide au médecin homéopathe dans le cadre du logiciel RADAR

Gillet, Jean-Christophe

Award date:
1988

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur .

Institut d'Informatique.

Année académique 1987 - 1988.

Implémentation d'une nouvelle méthode d'aide au médecin homéopathe dans le cadre du logiciel RADAR.

GILLET Jean-christophe.

Promoteur : J. Fichefet.

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Abstract.

Ce mémoire est consacré à l'implémentation d'une nouvelle méthode d'aide au médecin homéopathe. Cette implémentation s'intégre dans un logiciel plus vaste (RADAR) qui propose différentes méthodes d'aide au médecin homéopathe. La méthode développée ici est une méthode qui fait intervenir des algorithmes de théorie des graphes de façon à présenter au médecin un ensemble de remèdes possibles en fonction des symptômes qu'un patient présente. Le médecin choisira parmis ces remèdes le remède qui lui paraît le plus adéquat. Chaque méthode implémentée dans le logiciel RADAR présente des résultats devant aider le médecin homéopathe. Seule l'expérience sur des cas réels peut décider l'homéopathe à choisir une méthode plutôt qu'une autre.

Abstract.

This dissertation develops an implementation of a new method to help homeopaths in their work. This implementation is a part of computer system called RADAR (Rapid Aid to Drug Aimed Repertorization). That computer system proposes different kinds of help to the homeopath. The new method uses graphs theory and its algorithms to suggest a list of remedies to the homeopath. Those remedies are computed thanks to a list of symptoms introduced in the computer system. The homeopath will choose in that list of remedies the best remedy to prescribe to the patient. Each method is different but all the methods have the same purpose : helping the homeopath in his work. Only his own experience will make the homeopath choose one method instead of another.

Je tiens à remercier toutes les personnes qui m'ont permis de réaliser ce travail :

Monsieur J. Fichefet (promoteur du mémoire),

Monsieur J. Paris, pour son aide précieuse à la compréhension du logiciel RADAR,

Monsieur G. Bauduin, pour son aide lors de l'intégration,

Monsieur F. Schroyens, pour m'avoir fourni les cas cliniques nécessaires à la production de résultats.

Je voudrais également remercier tout particulièrement Messieurs G. Resconi et M. Trionfi pour leur accueil lors de mon séjour à Brescia.

TABLE DES MATIERES.

Page

CHAPITRE VI : RESULTATS ET INTERPRETATION.

INTRODUCTION.

1. Introduction générale.

Le terme "système d'aide" est un terme à la mode. Que ce soit dans le domaine économique, dans le domaine juridique, ou dans d'autres domaines, partout fleurissent ces systèmes d'aide qui doivent permettre de faciliter le processus de prise de décision.

Jusqu'il y a peu, le domaine médical était relativement épargné par ce genre de système. Les applications informatiques au niveau médical se limitaient à des applications de gestion essentiellement dans les hopitaux.

Actuellement, on en est arrivé à construire des systèmes quipermettent au médecin de gérer les dossiers de ses patients. Se sont également développés autour de cette relation entre un médecin et son patient toute une série de systèmes qui aident le médecin à poser son diagnostic médical. Ces logiciels existent tant dans pour le médecin généraliste que pour les médecins pratiquant les médecines dites parallèles, telle l'homéopathie.

L'idée de ce mémoire est d'implémenter, en homéopathie, une nouvelle méthode qui devrait aider le médecin homéopathe dans sa recherche d'un remède à prescrire à un patient en fonction de symptômes que celui-ci présente. Cette implémentation ne sera, en fait, qu'un ajout à un logiciel existant (avec tous les problèmes que cela pose).

2. Présentation des chapitres.

Chapitre I

Le contexte général de ce mémoire étant l'homéopathie, nous en rappellerons ici les grands principes ainsi que les outils qui aident le médecin homéopathe dans sa relation au patient.

Chapitre II

Une fois le programme réalisé, il s'agira de l'intégrer au logiciel existant : RADAR. Il est donc important de voir comment fonctionne ce logiciel et où viendra s'insérer le programme créé.

Chapitre III

Ce chapitre développe la partie théorique de la méthode à implémenter.

Chapitre IV

Nous y exposons les fonctionalités qui devraient être implémentées au travers d'une sorte de mode d'emploi prédéfini du programme.

Chapitre V

Ce chapitre décrit l'architecture du programme réalisé ainsi que les différentes fonctions du programme.

Chapitre VI

Nous verrons brièvement dans ce chapitre ce que nous obtenons comme résultats à partir de cas réels et comment nous pouvons interpréter ces résultats.

CHAPITRE I : Bases de l'homéopathie.

L'homéopathie est certainement, parmi les médecines parallèles, l'une des plus à la mode. Ses origines remontent à une époque où on en était aux balbutiements des soins médicaux. Actuellement, elle dispose de bases solides pour pouvoir s'exercer.

I.1 Les principes de l'homéopathie.

La loi de base qui régit l'homéopathie est la loi de similitude. Cette loi, déjà mentionnée par Hippocrate ne fut expérimentée de manière systématique qu'à partir de la fin du XVIIIème siècle par Samuel Hahnemann. Elle repose sur toute une série de propositions dont nous extrayons les trois suivantes :

proposition 1 :

> "Les maladies n'étant que des changements dans l'état général de l'homme, qui s'annoncent par des signes morbides, et la guérison n'étant possible non plus que par la conversion de l'état de maladie en celui de santé, on conçoit sans peine que les médicaments ne pourraient guérir les maladies s'ils n'avaient la faculté de changer l'état général de l'homme, consistant en sensations et actions, et que c'est uniquement sur cette faculté que repose leur vertu curative." [HAHN,paragr 19]

proposition 2 :

> "Il n'y a pas moyen de reconnaître en elle-même, par les seuls efforts de l'intelligence, cette faculté cachée dans l'essence intime des médicaments, cette aptitude virtuelle à modifier l'état du corps humain et par cela même à guérir les maladies. Par conséquent, il faut s'en tenir uniquement aux accidents morbides que les médicaments provoquent dans le corps sain, comme à la seule manifestation possible de la vertu curative dont ils jouissent, si l'on veut apprendre, à l'égard de chacun d'eux, quelles maladies il est en état de guérir." [HAHN, paragr 20 et 21]

proposition 3 :

> "Une affection dynamique, dans l'organisme vivant, est éteinte d'une manière durable par une plus forte, lorsque celle-ci, sans être de même espèce qu'elle, lui ressemble beaucoup quant à la manière dont elle se manifeste. La puissance curative des médicaments est donc fondée sur la propriété qu'ils ont de faire naître des symptômes semblables à ceux de la maladie et surpassant en force ces derniers." [HAHN,paragr 26 et 27]

En résumé, nous pouvons exprimer la loi de similitude de la manière suivante :

> Le médicament qui, administré à un patient, le guérira, est celui qui, administré à une personne saine aurait produit des symptômes similaires à ceux présentés par le patient.

I.2. Les outils de l'homéopathie.

Des expérimentations ont permis de récolter des pathogénésies, c'est-à-dire des listes de symptômes apparus chez un individu sain, à qui on a administré des substances ou des drogues. Ces pathogénésies sont regroupées dans différents ouvrages, outils de base du médecin homéopathe.

I.2.1. Les Matières Médicales.

Les Matières Médicales regroupent les ensembles des pathogénésies de différentes substances ou drogues classées alphabétiquement. Chaque pathogénésie recense un maximum de symptômes induits par la substance ou la drogue sur une personne saine. Ces symptômes sont classés alphabétiquement en 31 chapitres correspondant aux différentes parties du corps humain.

La plus connue de ces Matières Médicales est celle de T.F. Allen [ALLEN]. Cet ouvrage de 12 volumes recense plus de 1500 remèdes.

I.2.2. Les répertoires.

L'accès aux Matières Médicales sur base du remède étant fastidieux, certains auteurs ont eu l'idée de créer des répertoires dont l'accès se ferait sur base du symptôme. Dans ces répertoires, chaque symptôme référencie le ou les remèdes qui l'ont guéri ou qui le contiennent dans leur pathogénésie.

Le plus célèbre des répertoires est certainement celui de J.T. Kent [KENT]. Il est subdivisé en 31 chapitres suivant les différentes parties du corps humain (Mind, Vertigo, Head, Eye, ...). Chacun de ces 31 chapitres se compose de différentes rubriques correspondant à un symptôme général, classé par ordre alphabétique. Chaque rubrique contient :

- une liste générale des remèdes qui ont produit ce symptôme
- une liste des remèdes selon le moment de la journée.
- une liste des remèdes selon les circonstances d'aggravation de ce symptôme.

Chaque remède apparaissant dans une liste est affecté d'un degré. Ce degré est le degré d'efficacité (de 1 à 3) calculé en fonction de l'importance du remède constaté sur des cas cliniques ou lors d'expérimentation.

I.3. La démarche homéopathique. [RADAR]

La démarche du médecin homéopathe placé devant un patient comporte deux étapes :

- une individualisation du malade, c'est-à-dire une collecte de tous les symptômes qu'il présente
- une individualisation du remède, qui consiste à identifier la drogue (substance animale, minérale ou végétale) qui provoque chez une personne saine, les mêmes symptômes que ceux présentés par le malade. D'après la loi de similitude, cette drogue est censée guérir le malade.

Le médecin homéopathe dispose, pour individualiser ce remède, des différents outils décrits ci-dessus : Matières

Médicales et Répertoires. Une fois les symptômes collectés auprès du malade, il va, grâce à ces ouvrages tenter d'identifier le remède le plus adéquat à prescrire au malade. C'est pour aider le médecin homéopathe dans cette tâche que l'Unité de Recherche Archimède développe et commercialise depuis plusieurs années le logiciel RADAR (Rapid Aid to Drug Aimed Repertorization).

CHAPITRE II : Le logiciel RADAR.

II.1. But premier de RADAR.

Radar a pour but essentiel d'aider le médecin homéopathe dans sa recherche d'un remède à administrer à un patient présentant plusieurs symptômes. Pour cela, le logiciel offre d'une part un accès facile aux répertoires informatisés, d'autre part un traitement rapide d'un grand nombre de données, cherchant quel(s) remède(s) serai(en)t le(s) plus efficace(s) en fonction de symptômes désignés.

II.2. Aide à l'individualisation du malade.

La collecte des symptômes auprès du malade ne saurait en aucune façon être soutenue par un logiciel quelconque. Elle dépend essentiellement de l'expérience qu'a le médecin homéopathe pour interroger son patient et, bien entendu, de ce que le patient ressent.

L'identification de ces symptômes dans les répertoires dépend également en grande partie de l'expérience de l'homéopathe. Il s'agit ici de son expérience à manipuler les répertoires, de sa faculté à traduire des symptômes énoncés par le patient en langage de tous les jours en termes du répertoire. Cette recherche dans les répertoires peut toutefois être accélérée en facilitant l'accès aux répertoires. Le logiciel RADAR apporte ici son aide en permettant de naviguer très facilement dans les répertoires et en proposant une recherche par mots-clé.

II.3. Aide à l'individualisation du remède.

C'est surtout dans la recherche du "bon " remède que le logiciel RADAR apporte une aide précieuse au médecin homéopathe. Comment, en effet, identifier le bon remède lorsqu'on se trouve devant les listes de remèdes qui apparaissent en regard des symptômes sélectionnés par le médecin ? C'est à ce niveau-ci, par sa possibilité de traiter une grande quantité d'informations qu'un système comme RADAR va être utile. Le logiciel RADAR implémente plusieurs méthodes différentes pour traiter les listes de remèdes correspondant aux symptômes sélectionnés et suggérer des solutions au médecin.

II.3.1. Répertorisation classique.

La première méthode implémentée dans le logiciel RADAR consistait à présenter les résultats sous forme d'un tableau du type suivant :

	1	2	2	2	5	5	5	5	5	10	10	10	13	13	15	15
	acon.	ars.	calc.	puls.	nux-v.	phos.	plat.	rhus-t.	verat.	arn.	nat-m.	sep.	apis	camph.	agn.	aloe
	4/9	3/5	3/6	3/5	3/6	3/4	2/3	3/5	2/3	4/7	3/3	3/3	1/3	1/3	1/2	1/2
	2/6	2/4	2/4	2/4	2/3	2/3	2/3	2/3	2/3	2/2	2/2	2/2	1/3	1/3	1/2	1/2
1-	3	3	3	3	1	1	1	2	2	1	1	1	-	3	-	-
2-	3	1	1	1	2	2	2	1	1	1	1	1	3	-	2	2
3-	1	-	-	-	-	-	-	-	-	3	-	-	-	-	-	-
4a	-	1	2	1	3	1	-	2	-	2	1	1	-	-	-	-
5a	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Les cinq symptômes sélectionnés apparaissent au-dessus de ce tableau de résultats. La colonne qui précéde le numéro de symptôme indique le poids, c'est-à-dire l'importance, que nous avons attribué au symptôme. Dans le cas ci-dessus nous avons donné un poids un poids 0 aux trois derniers symptômes que nous considérons comme négligeables.

Voici décrit ligne par ligne le contenu de ce tableau :

- ligne 1 indique le rang du remède après analyse.
- ligne 2 reprend les noms des remèdes.
- ligne 3 indique le score général pour chaque remède :
    - le premier chiffre indique le nombre de symptômes couverts par le remède
    - le deuxième chiffre est la somme des degrés avec lesquels le remède intervient pour chaque symptôme.
- ligne 4 reprend un score alternatif pour chaque remède :
    - le premier chiffre indique le nombre de symptômes couverts par le remède mais en tenant compte des poids attribués aux symptômes.
    - le deuxième chiffre indique le total des degrés avec lesquels le remède intervient pour chaque symptôme mais en tenant compte des poids attribués aux symptômes.

- les cinq autres lignes reprennent pour chaque symptôme les remèdes qui interviennent pour ce symptôme dans le répertoire en indiquant leur degré.

Voilà, sans entrer dans les détails, les résultats que donne cette première méthode et donc une forme d'aide que le logiciel RADAR apporte au médecin homéopathe. Le traitement qui est effectué sur les données pourrait être effectué "à la main", mais il faudrait beaucoup d'effort pour arriver au même résultat.

II.3.2. Méthode Vithoulkas.

La méthode Vithoulkas implémentée par le programme VES dans le logiciel RADAR est un système de répertorisation qui possède les caractéristiques suivantes :

- tout comme n'importe quel système de répertorisation, il se base exclusivement sur les symptômes proposé par le médecin
- la phase de répertorisation proprement dite est constituée d'un système mathématique qui est basé tant sur les Répertoires et Matières Médicales que sur l'expérience clinique de Mr. Vithoulkas. C'est dire que le système contient tout un système de valorisation
    - des différents symptômes d'après leur localisation, ...
    - des remèdes suivant qu'ils sont polychrests (remèdes universels) ou non.

Pour des raisons de confidentialité exigées pour des raisons commerciales, il nous est malheureusement impossible d'en dire plus. Nous espérons que le lecteur comprendra aisément. Ajoutons néanmoins que les résultats que sort VES semblent appréciés ou retenir toute l'attention de médecins homéopathes même quand ils ne font pas partie de l'école de pensée Vithoulkas.

II.4. Conclusion.

On pourrait multiplier à l'infini les méthodes pour traiter les listes de remèdes. Les deux méthodes ci-dessus ne sont que deux exemples parmi d'autres. Le chapitre suivant présente dans le détail une nouvelle méthode dont l'implémentation est l'objet de ce mémoire.

CHAPITRE III : La méthode du remède central.

La méthode du remède central est proposée par G. Resconi et M. Trionfi ([1]). Ceux-ci proposent, sous certaines hypothèses, une formalisation du répertoire de Kent par l'introduction dans celui-ci d'une structure logique entre symptômes et remèdes. Cette approche nouvelle va permettre d'utiliser de façon simple la théorie des graphes dans la recherche du remède à proposert aumédecin en fonction de symptômes qu'il aura sélectionnés.

III.1. Rappel de théorie des graphes. [GRAPHE]

III.1.1. Définitions.

a) Graphe, ordre, sommet et arc.

Un **graphe** G(X,U) est constitué :

- d'un ensemble X = (x1, x2, ..., xn) fini ou dénombrable, dont les éléments sont les **sommets du graphe** et dont le cardinal $|X| = n$ est appelé **ordre** du graphe
- d'une famille U = (u1, u2, ..., un) d'éléments dans X*X, appelés les **arcs du graphe**.

b) Extrémités initiale et terminale d'un arc.

Soit un graphe G(X,U), avec X = (x1, x2, ..., xn).

Si $u = (xi, xj) \in U$, on dit que :

- xi est l'**extrémité initiale** (ou l'origine)
- xj est l'**extrémité terminale** (ou finale)

de l'arc $u \in U$.

c) Poids d'un arc.

Soit un graphe G(X,U).

Le **poids de l'arc** u = (xi,xj) $\in$ U est un nombre p(xi,xj) ou p(u) que nous associons à l'arc u.

d) Chemin, longueur d'un chemin et poids d'un chemin.

Soit un graphe G(X,U).

Un **chemin** est une suite u = (u1, u2, ..., ui, ui+1, ... uq) d'arcs telle que l'extrémité terminale d'un arc ui (avec i = 1, ..., q-1) coïncide avec l'extrémité initiale de l'arc suivant ui+1. Le nombre d'arcs qui composent la suite u est la **longueur du chemin** u.

Si x1 est l'extrémité initiale de u1 et si xq est l'extrémité terminale de uq, on dit que le chemin u joint x1 à xq.

Le **poids d'un chemin** est la somme des poids des arcs qui le composent.

e) Chemin élémentaire.

Un chemin est appelé **chemin élémentaire** lorsqu'il ne passe pas plus d'une fois par chacun des sommets du graphe.

f) Chemin minimal et distance.

Soit un graphe G(X,U).

Un chemin joignant xr $\in$ X à xi $\in$ X est dit **chemin minimal** si parmi tous les chemins joignant xr à xi, son poids total est minimal. La **distance** entre un sommet et un autre est la longueur du chemin minimal entre les deux sommets.

III.1.2. Algorithmes.

Les notions décrites ci-dessus s'appliquent à la catégorie des graphes orientés. Nous travaillerons, nous, uniquement avec des graphes non-orientés. Pour les graphes de type non orientés, on parlera d'arêtes au lieu d'arcs et le passage, dans un graphe, d'un sommet à un autre, s'ils sont reliés par une arête, pourra s'effectuer dans les deux sens et non plus dans le sens de l'arc.

Contrairement à la notation habituelle, nous utiliserons la notation ΓX pour désigner l'ensemble des sommets reliés aux sommets de l'ensemble X par une arête. Si Xi appartient à cet ensemble ΓX, c'est qu'il existe une arête joignant Xi à un sommet de X. Nous utiliserons également par la suite le terme de "sommets adjacents aux sommets de l'ensemble X".

a) Algorithme de Moore

Cet algorithme permet de trouver la longueur d'un chemin minimal d'un sommet à un autre.

La version suivante de l'algorithme trouve les longueurs des chemins minimaux d'un sommet xr à tous les autres.

```
k = 0
pi(xr) = 0
D = {xr}
S = {xr}

pour tout élément x appartenant à X \ {xr} effectuer
    | pi(x) = + ∞

tant que S <> 0 répéter
    | S = ΓS ∩ (X \ D)
    | k = k + 1
    | pour tout élément x appartenant à S effectuer
    |     | pi(x) = k
    | D = D U S
```

(version 1 de Moore)

- D est l'ensemble des sommets en cours de développement.

- S est l'ensemble des sommets déjà développés.

- pi(x) est la longueur du chemin minimal de xr à x, avec x appartenant à D.

- ΓS représente l'ensemble des sommets reliés aux sommets de S par une arête.

Une version modifiée de l'algorithme va nous permettre de trouver la longueur du chemin minimal d'un sommet xr à un autre xk et plus à tous les autres.

```
k = 0
D = {xr}
S = {xr}

tant que xk n'appartient pas à S

    | S = ΓS ∩ (X\D)
    | D = D U S
    | k = k + 1

d(xr,xk) = k
```

(version 2 de MOORE)

- D, S, ΓS : voir définition ci-dessus

- d(xr,xk) donnera la longueur du chemin minimal de xr à xk.

b) Distance minimale.

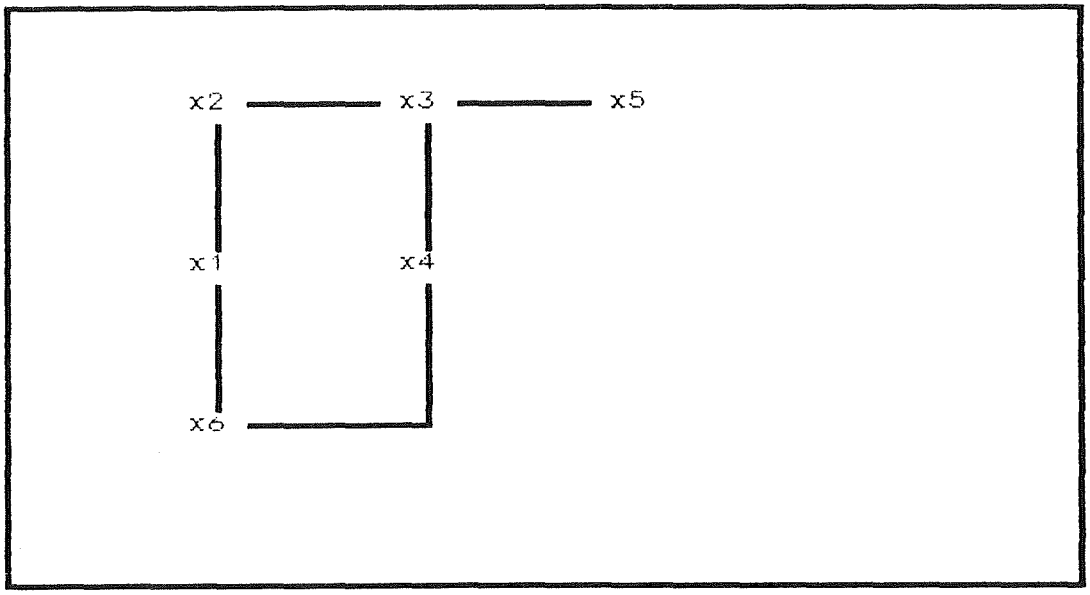
La distance minimale d'un sommet aux autres est la distance minimale nécessaire pour atteindre à partir de ce sommet tous les autres.

Sur un graphe G(X,U) avec X = {x1,x2, ..., xn} , on aura que :

$$\min \text{dist}(xr) = \max\left(\min\left(d(xr,xi)\right)\right) \quad \begin{matrix} xr \text{ appartenant à } X \\ i = 1, \ldots, n \end{matrix}$$

- min (d(xr,xi)) est donné par la version 2 de l'algorithme de Moore.

Cherchons par exemple sur le graphe suivant la distance minimale pour x1.



on a que :

- min (d (x1,x2)) = 1
- min (d (x1,x3)) = 2
- min (d (x1,x4)) = 2
- min (d (x1,x5)) = 3
- min (d (x1,x6)) = 1

donc :

$$\min \text{dist}\ (x1) = \max\ (\min\ (d\ (x1,xi)))\quad i = 1,\ldots 6$$
$$= 3$$

En 3 "coups" maximum, x1 peut atteindre tous les autres sommets du graphe.

La distance minimale peut aussi être définie comme la distance entre le sommet xr et le sommet le plus éloigné.

L'algorithme suivant, dérivé de l'algorithme de Moore permet de trouver directement la distance minimale du sommet xr.

```
k = 0
D = {xr}
S = {xr}

tant que S <> ∅

    | S = ΓS ∩ (X\D)
    | D = D U S
    | k = k + 1

min dist (xr) = k
```

(distance minimale)

- D est l'ensemble des sommets en cours de développement.
- S est l'ensemble des sommets déjà développés.
- min dist (xr) donnera la distance minimale pour le sommet xr.
- ΓS est l'ensemble des sommets adjacents aux sommets de l'ensemble S.

III.2. Théorie de l'équivalence logique.

III.2.1. Hypothèse générale.

G. Resconi et M. Trionfi ([RESCONI]) proposent d'appliquer au répertoire de Kent une structure logique, une équivalence logique entre remèdes et symptômes. Cela suppose bien sûr que l'on considère que :

toutes les expériences homéopathiques contenues dans le répertoire sont vraies.

(hypothèse générale)

C'est évidemment l'hypothèse que fait tout médecin homéopathe utilisant le répertoire.

III.2.2. Définition de l'équivalence logique.

En fonction d'un patient qui se présente en consultation, nous pouvons attribuer à chaque remède R du répertoire une valeur logique "vrai" ou "faux" suivant que ce remède lui est prescrit ou non. Nous aurons donc que pour un remède R particulier :

- R est vrai si le remède est prescrit au patient(noté R)
- R est faux si le remède ne lui est pas prescrit (noté $\overline{R}$).

De même, nous pouvons attribuer une valeur logique à chaque symptôme S du répertoire, suivant que ce symptôme est présent ou non chez le patient. Et, pour un symptôme particulier, nous aurons :

- S est vrai si le patient présente ce symptôme (noté S)
- S est faux si le patient ne le présente pas (noté $\overline{S}$).

Ces valeurs logiques sont attribuées lorsqu'on considère un patient particulier. Nous pouvons donc dire que c'est le patient qui, en se présentant chez le médecin donnera une valeur (vraie ou fausse) aux symptômes du répertoire en fonction des symptômes qu'il présente ou ne présente pas, tandis que c'est le médecin qui attribuera une valeur (vraie ou fausse) à un remède en le prescrivant ou non au patient.

Si on considère les expériences homéopathiques comme vraies (hypothèse générale) on peut établir entre un symptôme et un remède correspondant dans le répertoire de Kent la relation d'équivalence suivante :

$$R \Longleftrightarrow \overline{S}$$

Cette relation signifie qu'un remède R est logiquement équivalent à la négation du symptôme S. Cette équivalence logique peut être établie entre un symptôme et un remède si la correspondance existe dans le répertoire. Nous représenterons cette équivalence d'une autre manière sous forme de table, comme sur la page suivante.

	faux	vrai	R
faux	faux	vrai	
vrai	vrai	faux	
S			

III.2.3. Propriétés de l'équivalence logique.

a) Symétrie.

Lorsqu'on lit l'équivalence de gauche à droite (sens ==>), cette relation signifie que si le remède est présent alors le symptôme est absent. Si tel n'était pas le cas, cela voudrait dire que le répertoire de Kent comporte des erreurs. Or nous avons fait l'hypothèse que celui-ci était correct.

La lecture dans l'autre sens de l'équivalence (sens <==) est possible puisque l'équivalence dispose de la propriété de symétrie. Une lecture de droite à gauche signifie que si le symptôme est absent, le remède est prescrit.

Cette affirmation peut paraître assez bizarre à priori et ne peut être admise que si l'on considère l'hypothèse suivante que :

chaque symptôme résulte de l'absence dans l'organisme d'une matière physique qui a le même effet sur le symptôme S que le traitement R de la relation R <==> $\overline{S}$.

(hypothèse 2)

Ainsi, si le symptôme est absent cela signifie que cette matière physique, qui a le même effet que le remède, est présente dans l'organisme (dans le cas où le remède n'est pas prescrit).

Cette hypothèse est généralement acceptée par les médecins homéopathes qui considèrent que la santé n'est pas un phénomène passif mais actif. Il existe dans le corps des

remèdes qui assurent la santé. Chez une personne saine, tous les remèdes et les symptômes s'équilibrent. Lorsqu'un symptôme apparaît, un déséquilibre se crée. Un remède de l'extérieur (par opposition à un remède de l'intérieur) peut permettre l'élimination du symptôme et favoriser le retour du corps à l'équilibre. Les remèdes internes ne sont pas des remèdes contenus dans le répertoire de Kent mais ils leurs sont logiquement équivalents.

L'hypothèse 2 est nécessaire si nous voulons garder la relation d'équivalence entre un remède et la négation d'un symptôme. Sans cette hypothèse, nous n'aurions pas la symétrie et donc pas l'équivalence.

b) Transitivité.

L'équivalence possède également la propriété de transitivité. Cette propriété de transitivité va nous permettre de créer dans le répertoire un réseau reliant remèdes et symptômes.

La transitivité peut s'exprimer ainsi :

si A <==> B et si B <==> C alors A <==> C.

Appliqué aux remèdes et symptômes cela donne :

si R1 <==> $\overline{S1}$ et si $\overline{S1}$ <==> R2 et si R2 <==> $\overline{S2}$

alors R1 <==> $\overline{S2}$.

C'est-à-dire que, si :

- un remède R1 élimine un symptôme S1,
- un remède R2 élimine le même symptôme S1 et également un symptôme S2

alors :

- le remède R1 agit aussi sur le symptôme S2.

Remarquons que $\overline{S1}$ <==> R2 peut aussi s'écrire R2 <==> $\overline{S1}$ puisque nous avons la symétrie sur la relation d'équivalence.

Si on considère par exemple que :

- R1 représente le traitement "Bell"
- S1 représente le symptôme "Vertigo"
- R2 représente le traitement "Phos"
- S2 représente le symptôme "Looking"

Nous avons les relations logiques suivantes d'après le répertoire de Kent :

$R1 <==> \overline{S1}$ ou $Bell <==> \overline{Vertigo}$

$R2 <==> \overline{S1}$ ou $Phos <==> \overline{Vertigo}$

$R2 <==> \overline{S2}$ ou $Phos <==> \overline{Looking}$

et par la propriété de transitivité (et la propriété de symétrie pour pouvoir écrire $\overline{S1}$ <==> R2) nous pouvons écrire

$R1 <==> \overline{S2}$ ou $Bell <==> \overline{Looking}$

Donc, étant donné que Bell élimine Vertigo, que Vertigo est aussi éliminé par Phos qui élimine également Looking, le remède Bell a aussi une action sur le symptôme Looking. Cette relation entre "Bell" et "Looking" n'est pas mentionnée dans le répertoire mais elle peut en être déduite par transitivité. Les médecins homéopathes admettent généralement cette hypothèse que nous venons de faire que :

> un traitement agit non seulement sur les symptômes les plus proches (c'est-à-dire quand on utilise le répertoire sans tenir compte de la transitivité en ne considérant que la relation "tel symptôme est annulé par tel(s) remède(s)") mais aussi sur tous les symptômes liés à ce remède grâce à la propriété de transitivité. Cependant l'effet du remède sur le symptôme est moindre si la distance qui les sépare est grande.
>
> **(hypothèse 3)**

Remarquons que le médecin lui-même fait, sans s'en rendre compte ces déductions logiques lorsqu'il manipule le répertoire.

III.2.4. Conclusion.

L'équivalence logique munie des deux propriétés (symétrie et transitivité) va nous pemettre de considérer le répertoire de Kent comme dans un graphe ou un ensemble de graphes reliant les symptômes et les remèdes. L'exemple ci-dessus pourrait en effet se représenter sous la forme du graphe suivant :

Bell ----- Vertigo ----- Phos ----- Looking

Un sommet de ce graphe sera soit un remède soit un symptôme du répertoire de Kent. Une arête exprimera la relation du répertoire qui unit un remède au symptôme qu'il élimine. Le répertoire sera donc représenté sous forme d'un immense graphe dont nous ne considérerons qu'une partie en fonction des symptômes présentés par le patient.

III.3. La théorie des graphes dans la méthode.

Maintenant que nous avons vu comment grâce à l'équivalence logique établie entre un symptôme et un remède et grâce aux propriétés (symétrie et transitivité) de l'équivalence nous pouvions représenter le répertoire de Kent sous forme d'un immense graphe, voyons comment utiliser sur ce graphe la théorie des graphes. Nous utiliserons cette théorie pour effectuer certains traitements sur le graphe formé par le répertoire ou, plus précisément sur une partie de ce graphe construit à partir des symptômes effectivement présentés par un patient en particulier.

III.3.1. Construction du graphe symptômes-remèdes.

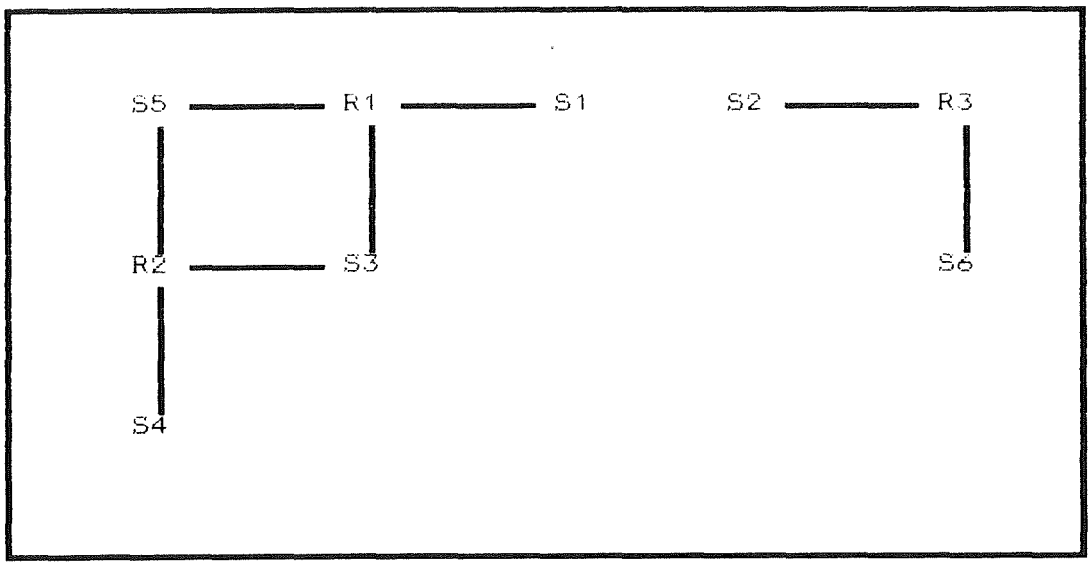
Après avoir collecté les principaux symptômes du patient, la première chose à faire sera de construire le graphe symptômes-remèdes du patient en utilisant le répertoire. Ainsi, si le patient présente les symptômes S1, S2, ..., Sn et si, par le répertoire, nous avons le même traitement R1 pour les symptômes S1 et S2, nous pourrons relié ces symptômes et ce remède de la façon suivante :

S1 ----- R1 ----- S2

Nous répéterons l'exercice pour tous les symptômes du patient et nous obtiendrons le graphe S-R du patient.

III.3.2. Distinction des graphes.

Une fois le graphe construit, rien ne dit s'il est constitué d'un seul ou de plusieurs morceaux (que j'appelerai ici sous-graphes). Si les symptômes S1, S2, S3, S4, S5 et S6 ont été sélectionnés, on pourrait avoir par exemple le graphe suivant :



Ce graphe est composé de deux sous-graphes.

Il serait utile de disposer d'un algorithme qui, connaissant les connections du graphe, pourrait nous dire s'il est composé de plusieurs sous-graphes et quels sont les sous-graphes. L'algorithme très simple de la page suivante permet de trouver tous les sommets du sous-graphe contenant un certain sommet Xr.

```
S = (Xr)
D = (Xr) U ΓS

tant que D <> S

    | S = D
    | D = ΓS
```

(distinction des graphes)

- ΓS représente l'ensemble des sommets adjacents aux sommets de l'ensemble S.

- D fournit, en fin d'algorithme, l'ensemble des sommets du sous-graphes contenant Xr.

Il suffira d'appliquer l'algorithme à un sommet ne faisant pas encore partie d'un sous-graphe exploré pour connaître tous les sommets du sous-graphe comprenant ce sommet.

Il est possible de tomber au cours de cette distinction des graphes sur un sous-graphe ne contenant qu'un seul sommet de type symptôme. Il s'agirait en fait d'un symptôme n'ayant, d'après le répertoire aucun remède commun avec les autres symptômes.

III.3.3. Algorithmes sur le graphe Symptômes-Remèdes (S-R).

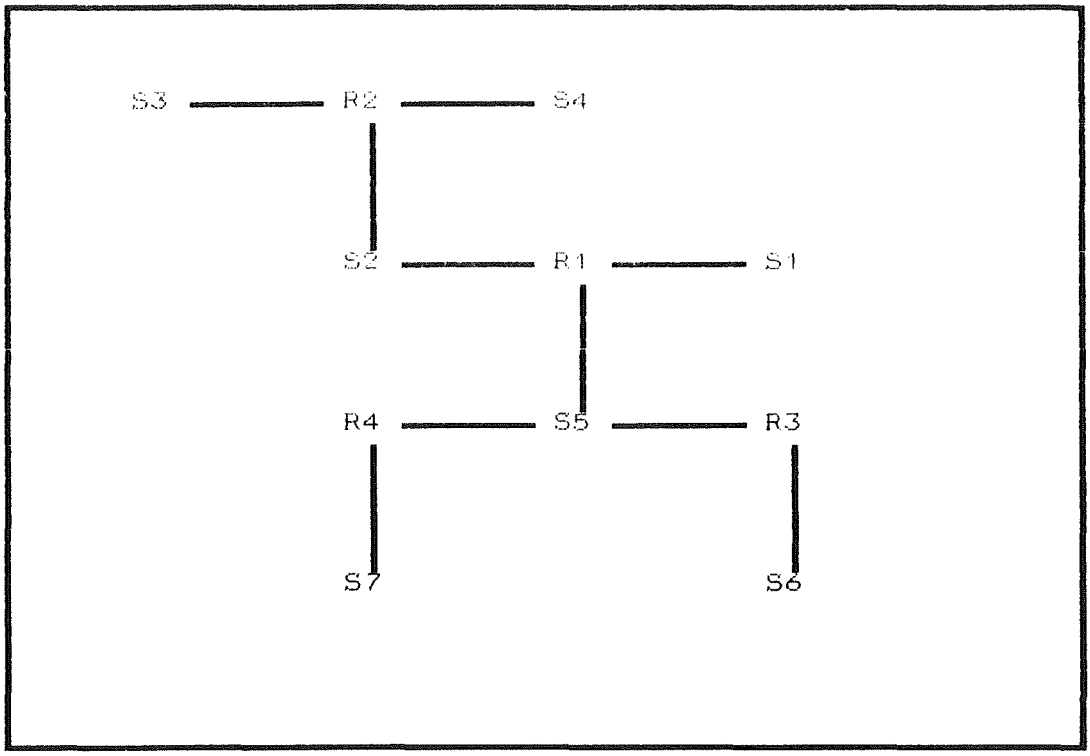
a) Le remède central.

Dans le graphe S-R (ou dans un sous-graphe S-R), le remède central est le sommet R de type remède pour lequel on a :

min (min dist (R))

avec R sommet de type remède et avec la distance minimale calculée entre ce R et tous les sommets de type symptôme. Le remède central est en fait le remède qui se trouve à une distance minimale par rapport à tous les symptômes.

Calculons par exemple le remède central du graphe de la page suivante.

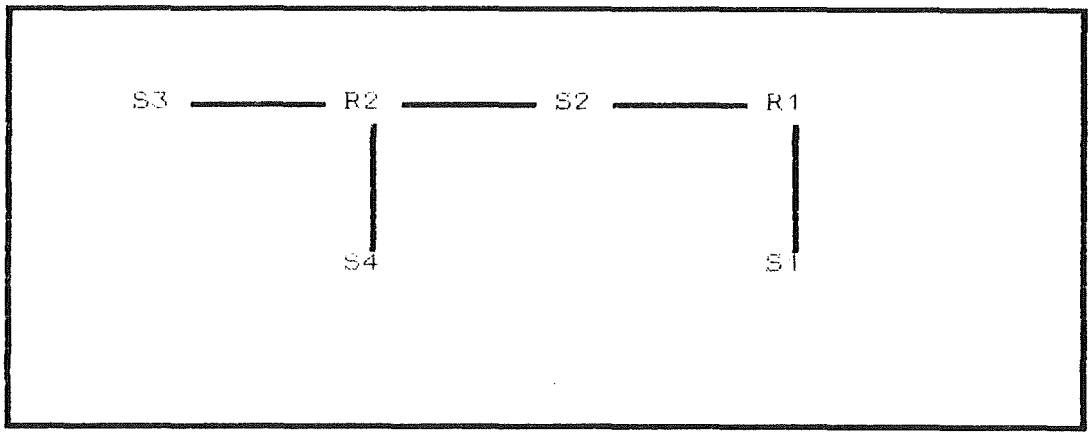


On cherche la distance minimale de chaque traitement. On obtient ainsi que :

- min dist (R1) = 3
- min dist (R2) = 5
- min dist (R3) = 5
- min dist (R4) = 5

Donc min (min dist (R)) est atteint pour R1.

Le remède central peut ne pas être unique comme sur le graphe de la page suivante.



- min dist (R1) = 3
- min dist (R2) = 3

et nous avons 2 remèdes centraux.

Il suffit de calculer pour chaque sommet de type R sa distance minimale (min dist R) par rapport aux sommets de type S par l'algorithme de la distance minimale et de déterminer le sommet R pour lequel on a min (min dist (R)).

b) Le rayon.

Le rayon est la distance entre le remède central et le plus éloigné des symptômes. C'est tout simplement la distance minimale pour le remède central.

Dans l'exemple 2 le rayon vaut 3.

III.4. Conclusion.

L'équivalence logique nous a donc permis de considérer le répertoire de Kent comme un immense graphe. Nous utiliserons les algorithmes de théorie des graphes pour effectuer des traitements sur une partie de ce graphe. Un certain nombre de fonctions seront proposées à l'utilisateur, dont la principale sera certainement le calcul du remède central qui

est censé être le remède le meilleur pour guérir le patient. Il se trouve en effet à une distance minimale par rapport à tous les symptômes du patient. Or, nous avons vu qu'un remède agit non seulement sur les symptômes les plus proches mais aussi sur les autres avec lesquels il est en relation via un chemin dans le graphe symptômes-remèdes du patient mais avec un effet moindre.

CHAPITRE IV : Fonctionalités à ajouter à RADAR.

La fonction principale de la méthode qui sera ajoutée au logiciel RADAR consiste bien entendu à présenter à l'utilisateur le remède central calculé à partir des symptômes sélectionnés. Outre cette fonction essentielle, en accord avec G. Resconi et M. trionfi, nous avons défini tout un ensemble de fonctions que devra remplir le logiciel. Nous en sommes ainsi arrivés à définir une sorte de mode d'emploi assez vague de la méthode. Ce mode d'emploi est bien entendu purement théorique. Il sert uniquement à clarifier les désirs de l'utilisateur. Il est évident qu'une implémentation avec les problèmes qu'elle pose (notamment les problèmes dûs à un environnement déjà existant) risque de ne pas être fidèle par rapport à ce mode d'emploi.

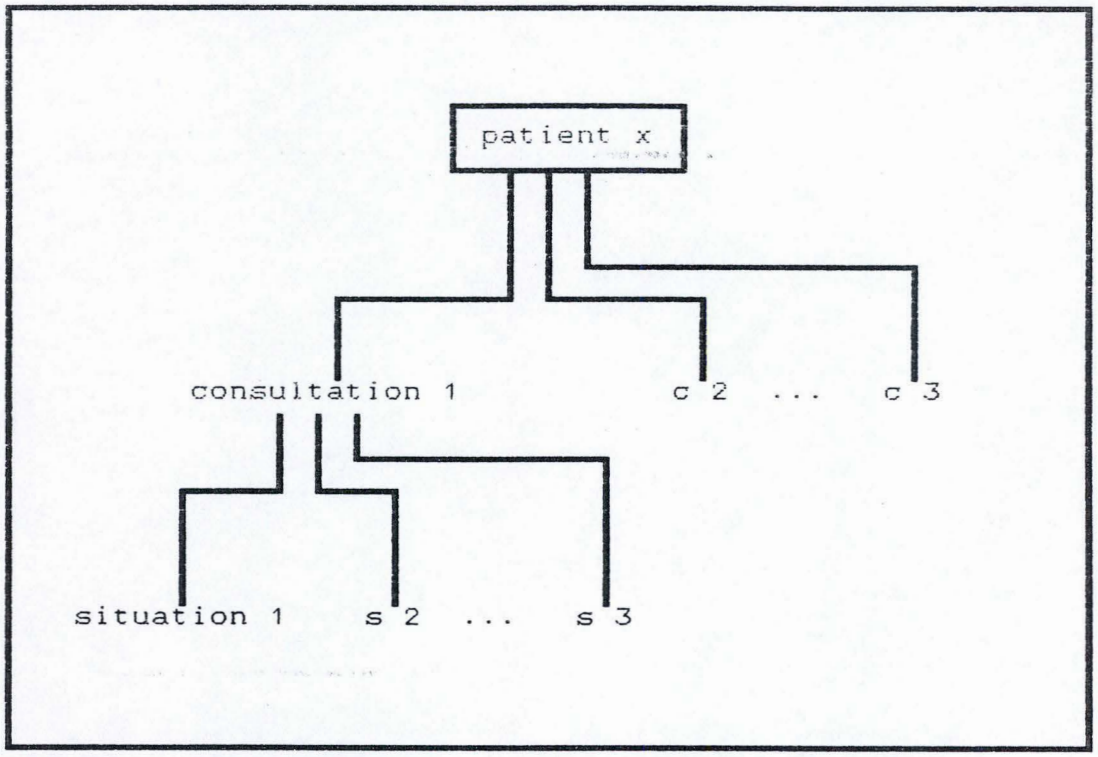
Nous énumérerons tout d'abord la liste des fonctions qui seront implémentées. Nous verrons ensuitse leur enchaînement dans un scénario traditionnel. Mais avant cela, nous allons réexpliquer la relation entre le patient et le médecin homéopathe.

IV.1. La relation patient-médecin.

Le médecin considère un patient au travers d'une consultation. Les consultations d'un même patient chez un médecin se succédent à intervalles irréguliers.

Au début d'une consultation, un patient présente un certain nombre de symptômes que le médecin homéopathe va collecter. Le médecin pourra, au cours de la consultation découvrir chez le patient d'autres symptômes ou supprimer certains symptômes qu'il juge secondaires. Ainsi, au cours d'une même consultation, nous distinguerons différentes situations, une situation étant caractérisée par un certain nombre de symptômes. A la fin de la consultation, le médecin prescrit un remède au patient.

Graphiquement, nous pourrions représenter un patient du point de vue du médecin sous forme de graphe. Le graphe de la page suivante montre cette relation patient-médecin.



Lorsque nous sommes dans une situation, c'est-à-dire lorsque nous avons différents symptômes, nous pouvons construire le graphe symptômes Remèdes de cette situation et pour ce graphe calculer le remède central. En fin de consultation, le médecin pourra, s'il le veut, prescrire le remède central de la situation finale ou bien un autre remède.

IV.2. Fonctionalités.

Lorsqu'on fait appel à la méthode, après avoir sélectionné un certain nombre de symptômes, il est possible de demander une des dix fonctions qui vont suivre. L'ensemble de ces fonctions constitue le menu de la méthode. L'objectif et le fonctionnement de chaque option sont décrits ici.

IV.2.1. Présentation du remède central.

a) Objectif.

Présentation pour la situation courrante du ou des remèdes centraux et des symptômes avec pour chaque symptôme la distance qui le sépare de son centre. Les symptômes sont présentés en ordre suivant la hiérarchie du répertoire de Kent.

b) Fonctionnement.

La présentation du remède central peut être demandée lorsqu'on se trouve dans une situation. Cette demande provoque l'affichage d'un écran du type suivant :

```
Remède central 1

Symptôme a            distance(Rc1,a)
Symptôme b            distance(Rc1,b)
  .                       .
  .                       .
  .                       .

Remède central 2

Symptôme a           distance(Rc2,a)
Symptôme b           distance(Rc2,b)
  .                       .
  .                       .
  .                       .



entrez une commande
```

Les différents remèdes centraux (s'il y en a plusieurs) sont présentés les uns derrières les autres. En dessous de chaque remède central, on affiche les symptômes de la situation en ordre hiérarchique suivant le répertoire de Kent, ainsi que la distance qui sépare chaque symptôme du remède central dans le graphe symptômes-remèdes de la situation courrante.

Dans le cas où le graphe de la situation se composerait de plusieurs sous-graphes, nous n'afficherons, en regard d'un remède central que les symptômes de ce sous-graphe et les distances entre ces symptômes et ce remède central dans ce sous-graphe.

IV.2.2. Suppression de symptômes.

a) Objectif.

Suppression d'un ou de plusieurs symptômes parmis les symptômes de la situation courrante, dans le cas où le médecin juge, par exemple, qu'un symptôme est secondaire. Cette suppression crée une nouvelle situation courrante.

b) Fonctionnement.

Lorsqu'on demande la suppression de symptôme(s), l'écran suivant apparaît :

```
1  Symptôme a
2  Symptôme b
3  Symptôme c
     .
     .
     .


entrez numéro de symptôme
```

Les symptômes de la situation courrante sont affichés. Ces symptômes sont numérotés. Grâce à cette numérotation, il est possible d'indiquer quel(s) symptôme(s) on désire supprimer. On sortira ensuite de la suppression pour revenir au menu de la méthode (d'où on pourra par exemple demander le remède central de la nouvelle situation ainsi créée).

IV.2.3. Annulation du remède central.

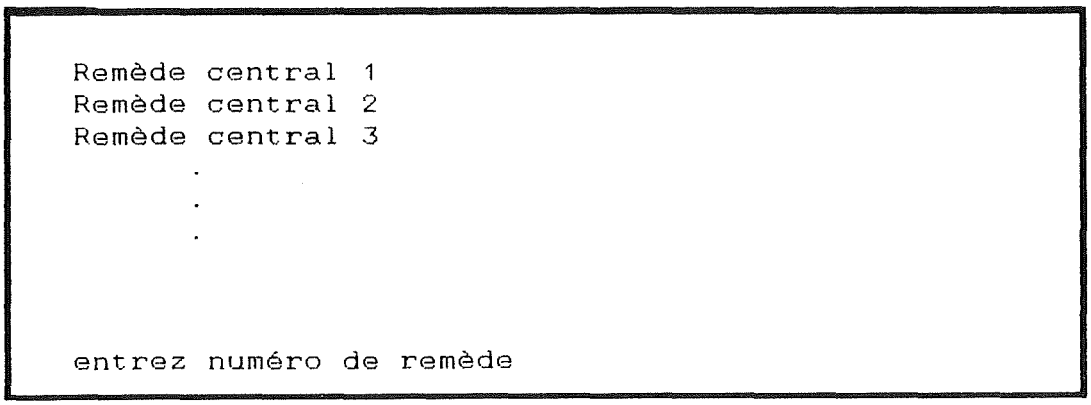
a) Objectif.

Possibilité, connaissant le(s) remède(s) central(aux) du graphe de la situation courrante de considérer ce graphe sans tenir compte du(d'un) remède central. C'est-à-dire qu'on considérera le graphe comme si le sommet remède central et les arètes qui y aboutissent n'existaient pas et on recalculera un autre remède central.

Ceci est utile dans le cas, par exemple, ou le médecin sait par expérience que le remède central a peu de chance d'être efficace compte tenu des symptômes présentés ou lorsqu'il est impossible de collecter de nouveaux symptômes (dans le cas d'un enfant, par exemple). Cette annulation crée une nouvelle situation courrante.

b) Fonctionnement.

Si on demande l'annulation du ou d'un remède central, l'écran suivant apparaît :



Le(s) remède(s) central(aux) de la situation courrante apparaîssent. Ces remèdes sont numérotés ce qui permet à l'utilisateur d'annuler celui (ou éventuellement ceux) qu'il désire. En général, on sortira ensuite de l'annulation pour revenir au menu de la méthode et demander le remède central de la nouvelle situation qu'on vient de créer.

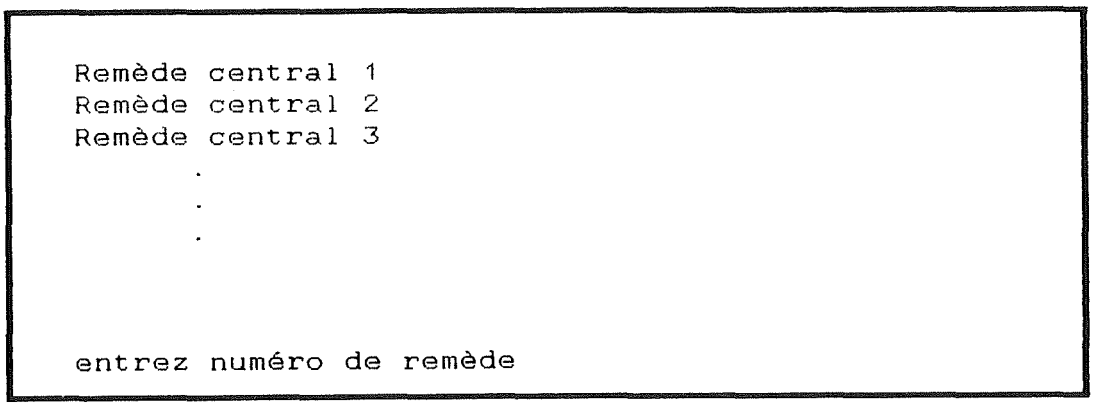
IV.2.4. Symptômes attachés au centre.

a) Objectif.

Connaître les symptômes attachés au remède central, c'est-à-dire les symptômes qu'élimine ce remède central d'après le répertoire, en précisant qu'on veut les symptômes de tel chapitre, telle rubrique, telle sous-rubrique, ... jusqu'au niveau de précision qu'on souhaite. Ces symptômes seront présentés accompagnés du degré d'efficacité sur le remède considéré.

b) Fonctionnement.

Lorsqu'on demande les symptômes attachés au remède central ou à un des remèdes centraux de la situation courrante, on voit apparaître l'écran suivant :



L'utilisateur peut choisir, parmis les remèdes affichés, le remède (ceux-ci sont numérotés) dont il veut connaître les symptômes. Une fois le remède choisi, l'utilisateur doit préciser dans quelle rubrique, sous-rubrique, ... il désire que la recherche s'effectue. L'écran résultat de la page suivante apparaît.

```
remède central

symptôme a
symptome b
symptôme c
    .
    .
    .


retour au menu général
```

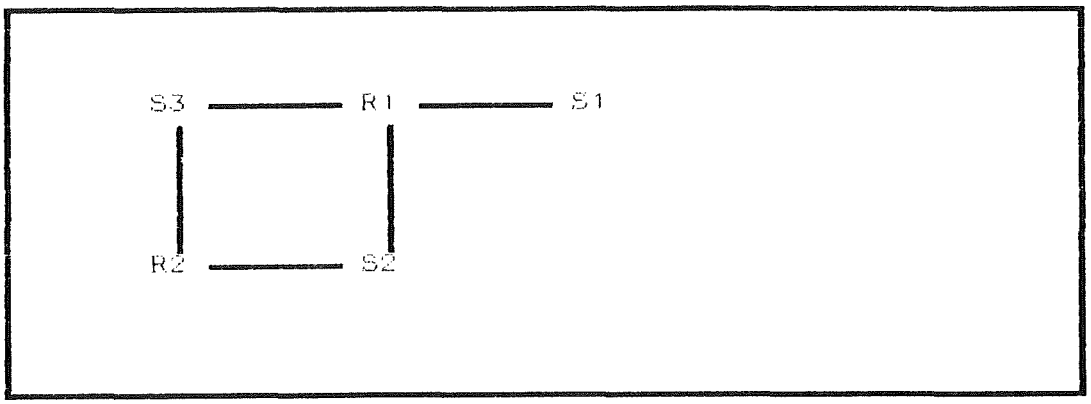
Cet écran présente les symptômes trouvés pour ce remède central. On sortira ensuite de cette fonction pour revenir au menu de la méthode.

IV.2.5. Suggestion de symptômes.

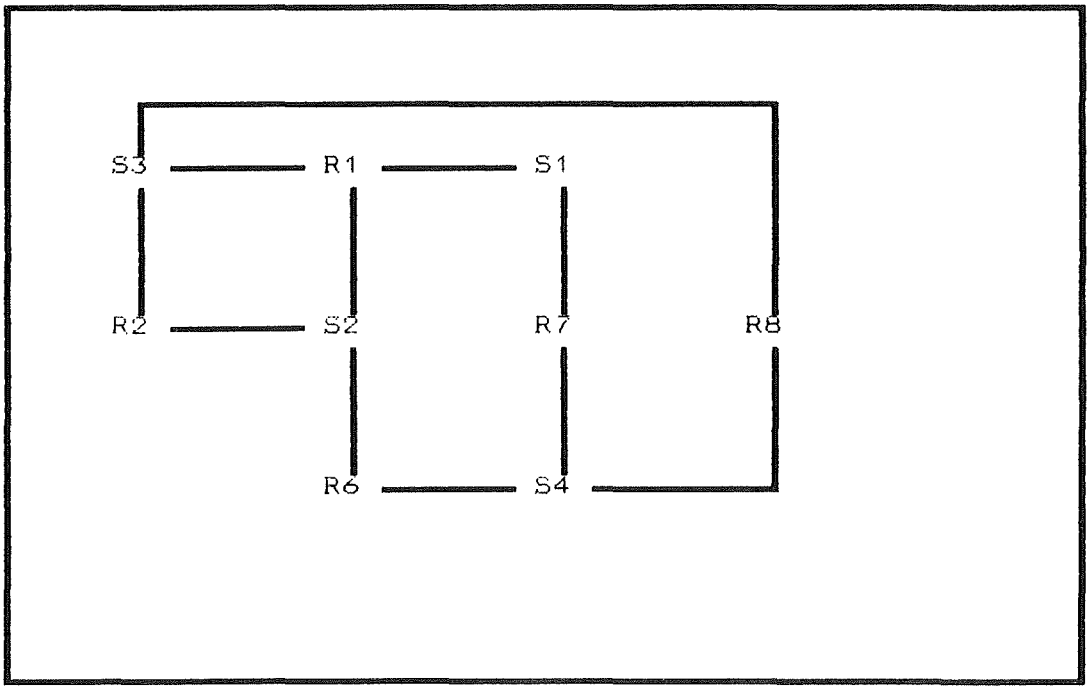
a) Objectif.

Demander des symptômes à l'intérieur d'un certain rayon à partir du remède central. Les premiers symptômes suggérés seront à distance 2 de tous les symptômes de la situation courrante (Ils auront en fait un remède commun avec chacun des symptômes). On pourra ensuite demander les symptômes à distance inférieure ou égale à 4 de tous les symptômes de la situation courrante, puis ceux à distance inférieure ou égale à 6 ... si on désire un éventail plus large.

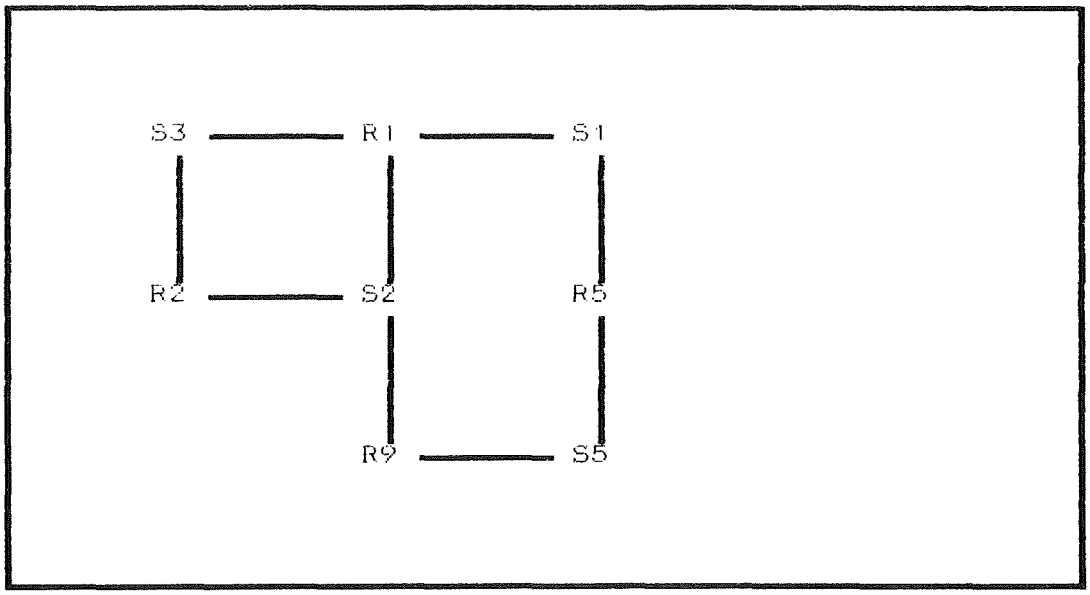
Par exemple, si on a pour la situation courrante les symptômes S1, S2 et S3, et que l'on arrive à former le graphe de la page suivante.



Après calcul, nous constatons que R1 est le remède central. Si on désire connaître les symptômes à distance 2 de tous les symptômes, on aura S4 comme symptôme suggéré si celui-ci est à distance 2 de S1, de S2 et de S3 par exemple de la façon suivante :



Si on veut les symptômes à distance inférieure ou égale à 4, on aura par exemple le graphe de la page suivante.



Le symptôme S5 est suggéré.

On précisera en plus à l'utilisateur quand les symptômes suggérés se trouvent en dehors du rayon de la situation courrante, ou plutôt à partir de quel moment on sort du rayon..

b) Fonctionnement.

Lorsqu'on demande une suggestion de symptômes, une liste des symptômes à distance 2 de tous les symptômes de la situation courrante apparaît dans un écran du type suivant :

```
Symptômes à distance 2

symptôme a      abréviation de chapitre   degré
symptôme b      abréviation de chapitre   degré
symptôme c      abréviation de chapitre   degré
   .
   .
   .

entrez commande
```

Sont affichés, en plus du libellé du symptôme, une abréviation du chapitre du répertoire dans lequel se trouve le symptôme (par exemple "m" pour Mind, "h" pour Head, ...) et le degré avec lequel le remède central apparaît pour ce symptôme dans le répertoire.

Il est ensuite possible de demander les symptômes à distance inférieure ou égale à 4 puis ceux à distance inférieure ou égale à 6 et ainsi de suite. Un écran du même type apparaîtra chaque fois. Lorsqu'on sortira du rayon du graphe de la situation courrante, la mention "hors du rayon" sera en plus affichée.

Enfin, on sortira de la suggestion de symptômes pour retourner au menu de la méthode.

IV.2.6. Mémorisation de la situation finale.

a) Objectif.

On désire mémoriser en fin de consultation des données suivantes concernant le patient :

- le nom du patient
- le prénom du patient
- l'âge du patient
- le sexe du patient
- les symptômes de la situation initiale
- les symptômes de la situation finale
- le remède central de la situation finale
- le remède prescrit effectivement
- le type du remède prescrit : il peut s'agir d'un remède différent du remède central ou bien du remède central ou bien du remède central après avoir annulé un remède central
- mode de prescription : par téléphone ou bien au cabinet

b) Fonctionnement.

La demande de mémorisation de la situation finiale provoque l'affichage d'un écran du type suivant qu'il suffit de remplir :

```
nom patient :
prénom patient :
âge :
sexe :

remède prescrit :
mode de prescription :



retour au menu général
```

On repassera ensuite au menu de la méthode.

IV.2.7. Histoire d'un patient.

a) Objectif.

Présentation de l'histoire d'un patient, c'est-à-dire des situations finales antérieures lors des consultations précédentes d'un patient. Il peut s'agir du patient en consultation ou bien d'un autre.

Pour chaque situation finale, on présentera le remède prescrit et les symptômes de cette situation finale. On indiquera en outre pour chaque situation finale antérieure la date de la prescription et si celle-ci s'est effectuée par téléphone. Et, pour chaque symptôme, on indiquera également s'il était présent lors de la situation finale précédente, s'il a été ajouté par rapport à la situation finale précédente ou s'il a été supprimé dans la situation finale suivante.

b) Fonctionnement.

Lorsqu'on demande l'histoire d'un patient, on doit tout d'abord préciser les nom et prénoms du patient ainsi que la date de la consultation dont on veut la situation finale. Pour cette date, on peut ne rien préciser et demander toutes les situations antérieures de ce patient. Apparaît alors le premier écran de cette histoire, c'est-à-dire l'écran de la première situation finale de ce patient ou bien l'écran de la situation finale à la date demandée (si on a précisé une date). Cet écran se présente ainsi :

```
nom       prénom          date          t

remède prescrit :

remède central :

symptôme a de situation finale
symptôme b de situation finale
symptôme c de situation finale
   .
   .
   .

entrez commande
```

"t" indique si la prescription s'est effectuée par téléphone.

L'utilisateur peut alors demander la situation finale de la consultation suivante (dans le cas où on n'a pas préciser la date) ou bien revenir au menu de la méthode.

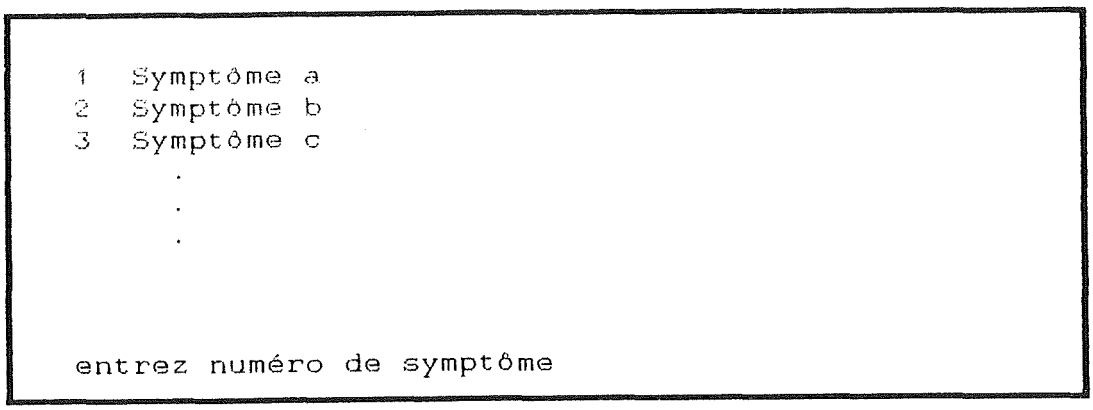
IV.2.8. Malades similaires.

a) Objectif.

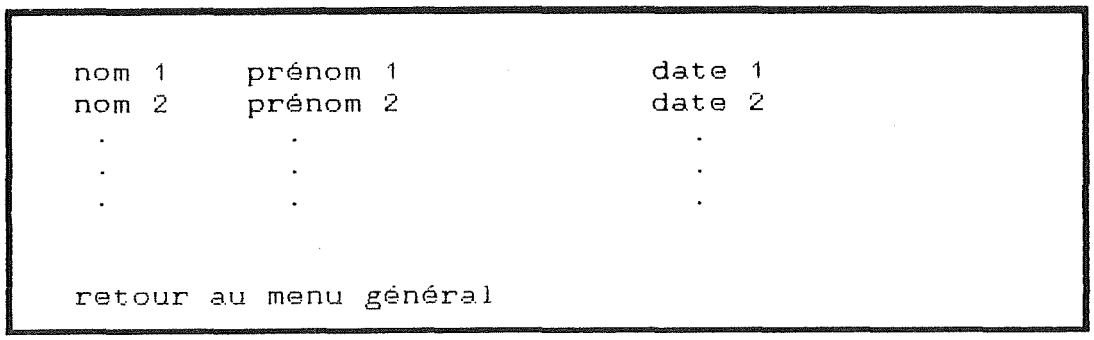
Demander les noms des malades qui présentaient en situation initiale tels symptômes communs avec le malade en cours de consultation. Ceci permet d'établir des analogies entre différents patients.

b) Fonctionnement.

Si on demande les malades similaires, un écran avec les symptômes initiaux du patient en consultation s'affiche :



On peut choisir dans cette liste un certain nombre de symptômes. Une fois ceux-ci choisis, on demandera quels sont les patients qui en situation initiale présentaient les mêmes symptômes que ceux qu'on vient de sélectionner. L'écran suivant apparaîtra :



On retournera ensuite au menu de la méthode, d'où on pourra par exemple demander l'histoire de ces malades similaires.

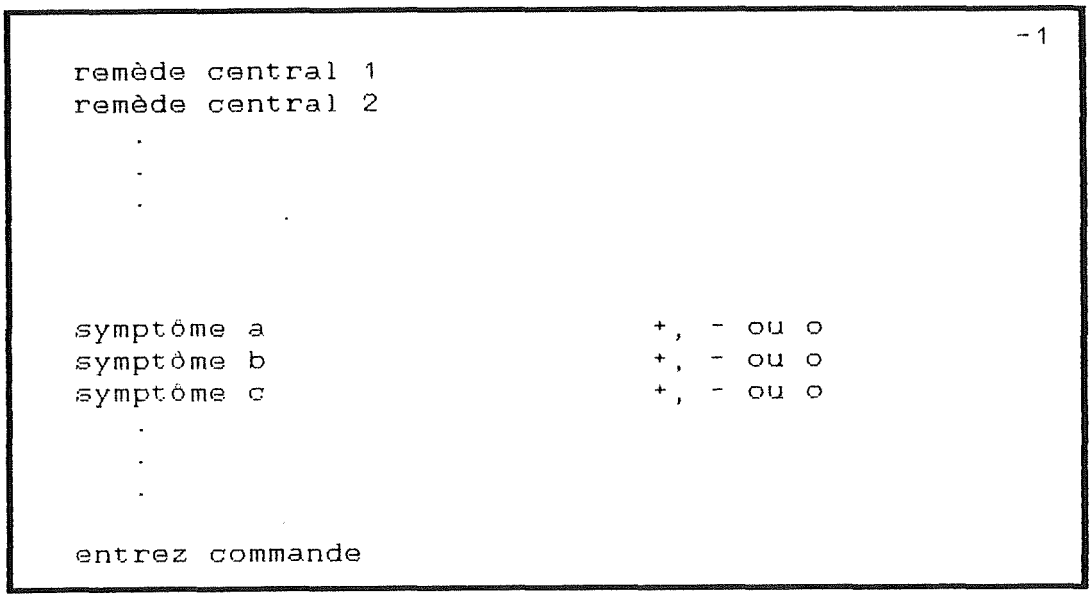
IV.2.9. Histoire de la consultation.

a) objectif.

Visionner les situations précédentes de la consultation courrante, c'est-à-dire voir quels étaient les symptômes et le remède central lors des situations antérieures. On indiquera pour chaque symptôme s'il était présent dans la situation précédente, s'il a été ajouté par rapport à la situation précédente ou s'il a été supprimer dans la situation suivante.

b) Fonctionnement.

Lorsqu'on demande la situation antérieure de la consultation courrante (si elle existe), l'écran suivant apparaît :



Le "-1" dans le coin supérieur droit indique la position de la situation affichée par rapport à la situation courrante. On peut ainsi remonter en -2, -3, ... tant qu'il y a des situations antérieures.

Pour chaque symptôme on indique :

- si le symptôme a été supprimé dans la situation suivant celle affichée

+ si le symptôme a été ajouté par rapport à la situation précédant celle affichée

o si le symptôme se trouve à la fois dans la liste des symptômes de la situation précédente et dans celle de la situation suivante.

IV.2.10. Retour à une situation antérieure.

a) Objectif.

On désire pouvoir considérer une situation antérieure, lorsqu'on visionne les situations antérieures de la consultation en cours, comme étant la nouvelle situation courrante. Ceci permettra entre autre de faire des essais (de se dire : "et si le patient avait ce symptôme en plus...") ou de corriger des erreurs (si on a introduit erronément un symptôme).

b) Fonctionnement.

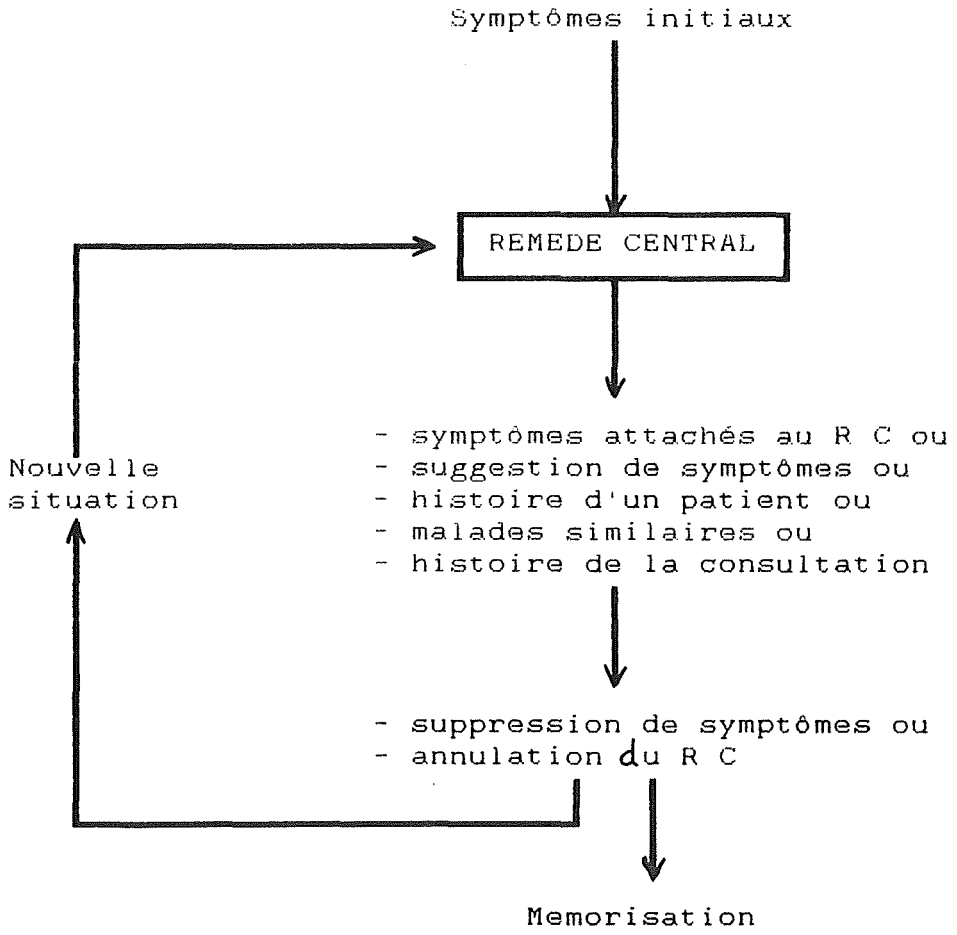
Il suffit de signaler quand on visionne les situations antérieures à la situation courrante que telle situation est maintenant la nouvelle situation courrante.

IV.2.11. Autres fonctions.

Il est évident que la liste des fonctions énoncées ci-dessus n'est pas exhaustive. On pourrait prévoir comme extension notamment une utilisation plus importante de toutes les données concernant les patients qui vont s'accumuler au cours du temps. La seule utilisation qui en est faite ici sert à trouver des cas similaires au patient en consultation. Il en existe d'autres.

IV.2. Enchaînement traditionnel des fonctions.

Une consultation traditionnelle par un médecin homéopathe utilisant le logiciel RADAR avec la méthode du Remède Central, et donc avec les fonctions décrites ci-desssus, peut être représentée de la manière suivante :



L'enchaînement se fait normalement ainsi :

1) Sélection dans le répertoire informatisé des symptômes que présente le patient.

2) Calcul et présentation du ou des remède(s) central(aux).

3) Demande des symptômes attachés au remède central (ou à un des remèdes centraux)
ou

demande de suggestion de symptômes

ou

demande de l'histoire d'un patient

ou

demande des malades similaires

ou

demande des situations antérieures de la consultation.

4) Suppression de symptôme(s) (avec ensuite retour en 2)

ou

annulation du ou d'un remède central (avec ensuite retour en 2)

ou

mémorisation de la situation finale

ou sortie.

Il s'agit ici d'un scénario traditionnel qui n'est évidemment pas imposé. C'est toutefois dans l'optique de réaliser un système appliquant ce scénario traditionnel que nous avons élaboré dans les grandes lignes le mode d'emploi des fonctions exposé ci-dessus.

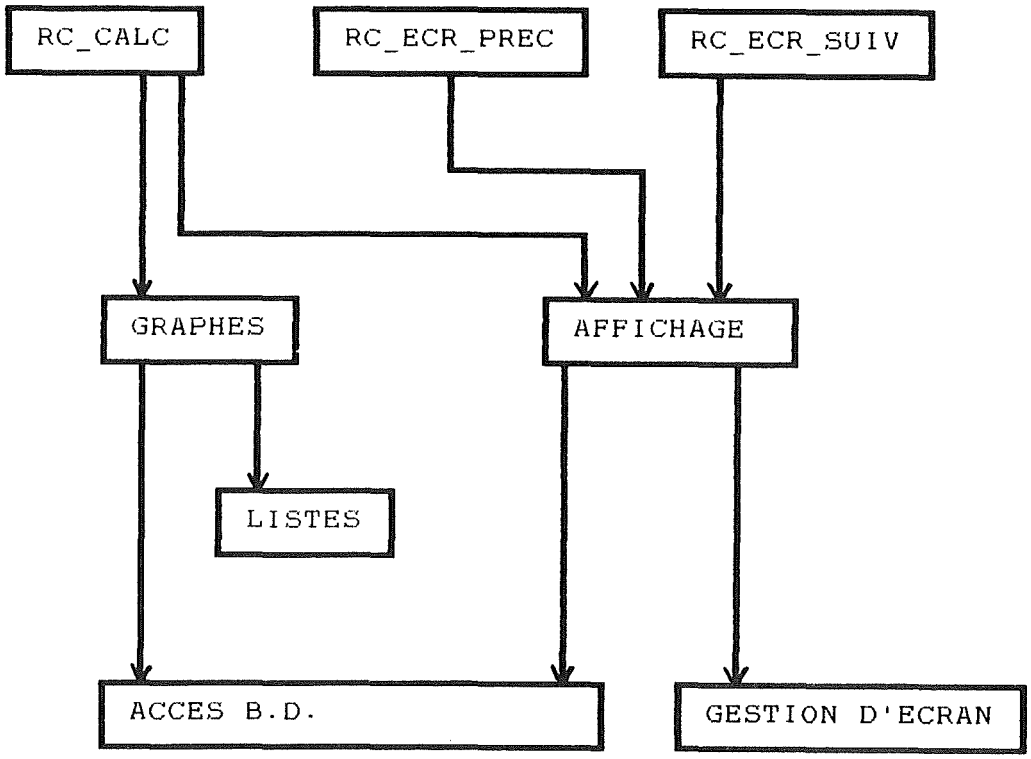
CHAPITRE V : Architecture et fonctionnement du programme.

V.1. Avertissement.

Le programme implémentant la méthode du remède central décrit ci-dessus devant s'intégrer dans un logiciel déjà conçu, nous disposons de toute une série de fonctions. Toutes les fonctions d'accès au répertoire de Kent existent de même que les fonctions permettant la présentation de données à l'écran. Par la suite, nous appellerons les premières fonctions base de données et les secondes fonctions d'écran.

V.2. Architecture.

V.2.1. Schema de l'architecture physique.



V.2.2. Routines par niveau.

niveau 4 : **fonctions utilisateurs**

rc_calc
rc_ecr_prec
rc_ecr_suiv

niveau 3 : a) **fonctions graphe**

gph_connec_fill
gph_simpl
gph_rc_all
gph_spt_isol
gph_ss_g
gph_rc_ss_g
gph_min_d
gph_adj
gph_sts
gph_an_co
gph_an_li
gph_lg_ch

b) **fonctions d'affichage**

aff_creer_tab_aff
aff_rc_display
aff_r
aff_s

niveau 2 : **fonctions liste**

lis_diff
lis_inter
lis_unio
lis_egali
lis_appart
lis_tri
lis_ajout

niveau 1 : a) **fonctions BD** (déjà disponibles)

b) **fonctions écran** (déjà disponibles)

V.3. Logique du programme.

Lorsqu'on fait appel à la méthode du remède central, c'est-à-dire à la fonction rc, on a déjà sélectionné un certain nombre de symptômes. Cette fonction va présenter un menu contenant différentes options, dont la présentation du remède central.

Le calcul du remède central (fonction rc_calc) se fera grâce aux différentes fonctions de travail sur les graphes (fonctions gph_).

L'affichage du remède central (fonctions rc_calc, rc_ecr_prec et rc_ecr_suiv) se fera via les fonctions d'affichages (fonctions aff_) qui permettent de créer la table d'affichage (aff) et d'afficher un écran de cette table d'affichage.

Pour permettre le fonctionnement des fonctions de travail sur le graphe (fonctions gph_) nous avons créé un ensemble de fonctions de gestion de listes (fonctions lis_) qui facilitent le traitement des listes de sommets du graphe.

Enfin, les fonctions graphes (notamment pour la construction du graphe) et les fonctions d'affichage feront appel à des fonctions d'accès à la BD c'est-à-dire à des fonctions d'accès au répertoire de Kent informatisé, qui existent déjà dans le logiciel RADAR. Les fonctions de gestion d'écran utiles pour l'affichage (aff_) existent également.

V.4. Spécification des fonctions.

Etant donné que tout l'environnement dans lequel va venir s'intégrer la nouvelle méthode est programmé en langage C, nous avons bien entendu conservé ce même langage.

V.4.1. Structure de données globale.

elist : structure chaînée contenant la description d'un sommet du graphe.

elist est composé de :

- ident : identificateur du sommet
- type : type du sommet :

  "S" pour un sommet de type symptôme

  "R" pour un sommet de type remède
- next : pointeur vers le sommet suivant de la liste.

V.4.2. Variables globales du programme.

CONNEC : matrice de connection du graphe de dimension

DIM_RC * NBR_MED.

(NBR_MED est le nombre de remèdes recensés dans le répertoire de Kent)

(DIM_RC est le nombre maximum de symptômes pouvant être sélectionnés)

ss_g : tableau de pointeurs vers les listes de sommets des différents sous-graphes.

Un élément de ce tableau correspond à un sous-graphe.

r_c : tableau de pointeurs vers les listes par sous-graphe de remèdes centraux.

r_c[i] pointe vers la liste des remèdes centraux du sous_graphe dont la liste des sommets se trouve en ss_g[i].

aff : tableau de pointeurs vers les listes d'affichage par écran.

aff[i] pointe vers la liste des sommets à afficher sur un même écran. Cette liste comporte au maximum 20 sommets de type "S".

isol_s : pointeur vers la liste des symptômes isolés.

l_rc : pointeur vers la liste des remèdes centraux d'un sous-graphe.

Ce pointeur sert à transmettre la liste des remèdes centraux de gph_rc_ss_g à gph_rc_all.

p_der : pointeur qui permet l'ajout d'un élément à une liste.

nbr_ss_g : nombre de sous-graphes composant le graphe dont la matrice de connection est CONNEC.

rc_done : booléen qui indique si le calcul du remède central a déjà été effectué.

nbr_rc : nombre de remèdes centraux du graphe dont la matrice de connection est CONNEC.

nbr_spt_isol : nombre de symptômes isolés du graphe dont la matrice de connection est CONNEC.

V.4.3. Variables globales du logiciel RADAR.

Ne sont reprises ci-dessous que les variables globales de RADAR utiles au fonctionnement de la méthode.

nbr_spt_sel : nombre de symptômes sélectionnés.

hst : table de longueur DIM_HST (DIM_HST est le nombre maximum de symptômes sélectionnables) qui garde trace des symptômes déjà sélectionnés.

Un élément de ce tableau contient notamment :

- adr_rub : adresse qui permet de retrouver l'intitulé du symptôme
- adr_rmd : adresse qui permet de retrouver la liste des codes remèdes pour ce symptôme

- lgr_rmd : longueur de la liste des remèdes.

V.4.4. Spécification par fonction.

Pour chaque fonction , nous définissons notamment ses arguments et ses résultats. Dans ces deux rubriques, nous mentionnons, outre les données passées en paramètre, dans un sens ou dans l'autre, les variables globales à la méthode qui interviennent en tant que données en entrée ou données en sortie dans la réalisation de la fonction.

1. rc_calc

Objectif : calcul et affichage du ou des remède(s) central(aux)

Argument : - CONNEC (variable globale à la méthode)

Pre : - CONNEC représente les connections du graphe de la situation courrante

Resultats :
- affichage du premier écran résultat
- aff (variable globale à la méthode)
- ss_g (id.)
- r_c (id.)
- isol_s (id.)
- nbr_rc (id.)
- nbr_spt_isol (id.)
- nbr_ss_g (id.)

Post :
- ss_g contient les listes des sommets des différents sous-graphes
- r_c contient pour chaque sous-graphe la liste des remèdes centraux. r_c[i] contient la liste des remèdes centraux du sous-graphe contenu en ss_g[i]
- isol_s contient la liste des symptômes isolés
- aff contient les listes d'affichage par écran
- nbr_rc contient le nombre total de remèdes centraux du graphe
- nbr_spt_isol contient le nombre de symptômes isolés du graphe
- 0 <= nbr_spt_isol <= nbr_spt_sel
- nbr_ss_g contient le nombre de sous-graphes
- 1 <= nbr_ss_g <= nbr_spt_sel
- l'écran affiché est contenu en aff[0]

Appelé de : rc

Appelle :
gph_connec_fill
gph_simpl
gph_rc_all
gph_spt_isol
aff_creer_tab_aff
aff_rc_display

Fonctionnement : après construction, le graphe est simplifié. On calcule le(s) remède(s) central(aux) de ce graphe. On vérifie ensuite s'il y a des symptômes isolé. Enfin, on crée la table d'affichage et on affiche le premier écran de cette table.

2. rc_ecr_prec

Objectif : affichage de l'écran de résultats précédent l'écran pno_ecr

Argument : - pno_ecr : numéro d'ecran

Pre : - 0 <= pno_ecr < DIM_AFF

Resultats : - pno_ecr
- ecran pno_ecr-1 ou message d'erreur affiché

Post : - si pno_ecr <> 0
l'écran affiché est celui contenu en aff[pno_ecr-1] et pno_ecr vaut pno_ecr -1
sinon
un message d'erreur est affiché signifiant qu'il n'y a pas d'écran précédent

Appelé de : rc

Appelle : aff_rc_display
fonctions d'écran

Fonctionnement : s'il existe un écran précédent, on fait appel à la fonction aff_rc_display pour afficher cet écran et on met pno-ecr à jour. Sinon, on affiche un message d'erreur grâce aux fonctions d'écran.

3. rc_ecr_suiv

Objectif : affichage de l'écran de résultats suivant l'écran pno_ecr

Argument : - pno_ecr : numéro d'écran

Pre : - 0 <= pno_ecr < DIM_AFF

Resultats : - pno_ecr
- écran pno_ecr+1 ou message d'erreur affiché

Post : - si pno_ecr+1 existe
l'écran affiché est celui contenu en aff[pno_ecr+1] et pno_ecr vaut pno_ecr+1 sinon un message d'erreur est affiché signifiant qu'il n'y a pas d'écran suivant.

Appelé de : rc

Appelle : aff_rc_display
fonctions d'écran

Fonctionnement : s'il esiste un écran suivant, on fait appel à la fonction aff_rc_display et on met pno_ecr à jour. Sinon, on affiche un message d'erreur grâce aux fonctions d'écran.

4. gph_connec_fill

Objectif : remplissage de la matrice CONNEC à partir des symptômes contenus dans hst

Argument :
- hst (variable globale à RADAR)
- nbr_spt_sel (variable globale à RADAR)

Pre :
- nbr_spt_sel > 1
- hst contient la description des nbr_spt_sel symptômes sélectionnés

Resultats :
- CONNEC (variable globale à la méthode du remède central)

Post :
- CONNEC(i,j) = 1 si j correspond à un code remède appartenant à la liste des codes remèdes du symptôme i

  CONNEC(i,j) = 0 sinon

Appelé de : rc_calc

Appelle : fonctions base de données

Fonctionnement : pour chaque symptôme i de hst, on va chercher les codes remèdes correspondant et on met à 1 l'élément de la matrice dont les coordonnées sont :

- i pour le numéro de ligne
- le code remède pour le numéro de colonne

5. gph_simpl

Objectif : simplification du graphe en mettant à 0 les colonnes qui ne contiennent qu'un seul 1.

Argument : - CONNEC (variable globale de la méthode)

Pre : - CONNEC représente les connections du graphe de la situation courrante

Resultats : - CONNEC : matrice de connection du graphe (variable globale de la méthode)

Post : CONNEC ne contient pas de colonne ne comportant qu'un seul 1

Appelé de : rc_calc

Appelle : /

Fonctionnement : on balaie une à une les colonnes de CONNEC. Si une colonne ne contient qu'un seul 1 cet élément valant 1 est mis à 0.

6. gph_rc_all

Objectif : calcul du(des) remède(s) central(aux)

Argument : - CONNEC (variable globale de la méthode)

Pre : - CONNEC représente les connections du graphe de la situation courrante

Resultats : - ss_g (variable globale de la méthode)
- r_c (variable globale à la méthode)
- nbr_rc (variable globale à la mèthode)

Post : - ss_g contient les listes des sommets des différents sous-graphes.
- r_c contient pour chaque sous-graphe la liste des remèdes centraux. r_c[i] contient la liste des remèdes centraux du sous-graphe contenu en ss_g[i].
- nbr_rc contient le nombre total de remèdes centraux du graphe.

Appelé de : rc_calc

Appelle : gph_ss_g
gph_rc_ss_g

Fonctionnement : après distinction des différents sous-graphes par la fonction gph_ss_g, on calcule pour chaque sous_graphe ses remèdes centraux grâce à la fonction gph_rc_ss_g.

7. gph_spt_isol

Objectif : donner une liste des symptômes isolés du graphe, c'est-à-dire une liste des symptômes qui n'ont aucun remède commun avec aucun autre symptômes.

Argument : - CONNEC (variable globale de la méthode)

Pre : - CONNEC représente les connections du graphe de la situation courrante

Resultats : - isol_s (variable globale de la méthode)

Post : - isol_s contient la liste des sommets de type S du graphe qui n'ont aucune connection avec des sommets de type R.
(i,"S") appartient à isol_s
si CONNEC(i,j) = 0 pour tout j
(avec i tel que 0 <= i < nbr_spt_sel)

Appelé de : rc_calc

Appelle : /

Fonctionnement : pour chaque symptôme sélectionné, on balaie la ligne correspondante dans la matrice de connection CONNEC. Si une ligne i ne contient que des 0, le symptôme i est isolé.

8. gph_ss_g

Objectif : distinction des différents sous-graphes du graphe représenté par la matrice de connection CONNEC.

Argument : - CONNEC (variable globale de la méthode)

Pre : - CONNEC représente les connections du graphe de la situation courrante

Resultats : - ss_g (variable globale de la méthode)
- nbr_ss_g (variable globale de la méthode)

Post : - ss_g contient les listes de sommets des différents sous-graphes
- nbr_ss_g contient le nombre de sous-graphes
- 1 <= nbr_ss_g <= nbr_spt_sel

Appelé de : gph_rc_all

Appelle : lis_diff
gph_sts
gph_adj
lis_unio
lis_egali
lis_tri

Fonctionnement : cette fonction applique de manière itérative l'algorithme de distinction des graphe décrit au chapitre IV. Tant qu'il y a un sommet qui ne fait pas partie d'un sous-graphe, on applique l'algorithme à ce sommet qui n'a pu encore être placé dans un sous-graphe.

9. gph_rc_ss_g

Objectif : calcul du(des) remède(s) central(aux) du sous-graphe ss_gph

Argument : - ss_gph : liste des sommets du sous-graphe
- CONNEC (variable globale de la méthode)

Pre : - ss_gph contient au moins trois sommets
- CONNEC représente les connections du graphe de la situation courrante.

Resultats : - l_rc (variable globale de la méthode)

Post : - l_rc est non vide

- l_rc contient la liste du(des) remède(s) central(aux) du sous-graphe ss_gph.

Appelé de : gph_rc_all

Appelle : gph_min_d

Fonctionnement : pour chaque sommet de type remède du sous-graphe, on calcule sa distance minimale. On ajoute ensuite à la liste l_rc tous les sommets de type remède pour lesquels cette distance minimale est minimale.

10. gph_min_d

Objectif : calcul de la distance minimale pour le sommet identifié par id_r dans le sous-graphe ss_gph.

Argument : - id_r : identifie le remède
- ss_gph : liste des sommets du sous-graphe

Pre : - 0 <= id_r < NBR_MED
- ss_gph contient au moins trois sommets
- (id_r,"R") appartient à la liste ss_gph

Resultats : - distance minimale du sommet id_r

Post : - 0 < distance minimale < nbr_spt_sel
- distance minimale est impair
- distance minimale est la distance minimale du sommet id_r dans le sous-graphe ss_gph

Appelé de : gph_rc_ss_g

Appelle : lis_diff
gph_adj
lis_inter
lis_unio

Fonctionnement : on applique l'algorithme de calcul de la distance minimale exposé au chapitre IV.

11. gph_adj

Objectif : donner une liste des sommets adjacents aux sommets de la liste p_lis

Argument :
- p_lis : liste de sommets
- CONNEC (variable globele de la méthode)

Pre :
- p_lis est non-vide
- CONNEC représente les connections du graphe de la situation courrante

Resultats :
- p_res (variable globale de la méthode)

Post :
- (id_r,R) appartient à p_res s'il existe (i,S) appartenant à p_lis tel que CONNEC(i,id_r) = 1
- (id_s,S) appartient à p_res s'il existe (j,R) appartenant à p_lis tel que CONNEC(id_s,j) = 1

Appelé de :
gph_min_d
gph_ss_g
gph_lg_ch

Appelle :
gph_an_co
gph_an_li

Fonctionnement : pour chaque sommet de la liste p_lis, on ajoute les sommets adjacents au sommet à la liste résultat sauf ceux qui font déjà partie de cette liste résultat.

12. **gph_sts**

Objectif : donner une liste de tous les sommets du graphe dont la matrice de connection est la matrice CONNEC

Argument :
- CONNEC (variable globale de la méthode)
- nbr_spt_sel (variable globale à RADAR)

Pre :
- CONNEC représente les connections du graphe de la situation courrante
- 1 < nbr_spt_sel

Resultats :
- p_res (variable globale de la méthode)

Post :
- nbr_spt_sel <= nombre de symptômes de p_res
- p_res contient la liste de tous les sommets du graphe dont CONNEC est la matrice de connection

Appelé de : gph_ss_g

Appelle : /

Fonctionnement : on ajoute a la liste p_res autant d'élément qu'il y a de symptômes c'est-à-dire nbr_spt_sel-1. On balaie ensuite la matrice CONNEC pour ajouter à la liste les remèdes du graphe.

13. **gph_an_co**

Objectif : mise à 0 de la colonne j de la matrice CONNEC.

Argument : - j : indice colonne
- CONNEC (variable globale de la méthode)

Pre : - CONNEC représente les connections du graphe de la situation courrante
- 0 <= j < NBR_MED

Resultats : - CONNEC (variable globale de la méthode)

Post : - CONNEC(i,j) = 0 pour tout i

Appelé de : gph_adj

Appelle : /

14. **gph_an_li**

Objectif : mise à 0 de la ligne i de la matrice CONNEC.

Argument : - i : indice ligne
- CONNEC (variable globale de la méthode)

Pre : - 0 <= i < DIM_RC
- CONNEC représente les connections du graphe de la situation courrante

Resultats : - CONNEC (variable globale de la méthode)

Post : - CONNEC(i,j) = 0 pour tout j

Appelé de : gph_adj

Appelle : /

15. gph_lg_ch

Objectif : calcul de la longueur du chemin minimal entre le sommet id_r de type remède et le sommet id_s de type symptôme dans le sous-graphe ss_gph.

Argument :
- id_r : identifie le sommet de type remède
- id_s : identifie le sommet de type symptôme
- ss_gph : liste des sommets du sous_graphe

Pre :
- 0 <= id_r < NBR_MED
- 0 <= id_s < nbr_spt_sel
- id_s et id_r appartiennent à ss_gph
- ss_gph contient au moins trois sommets

Resultats :
- longueur

Post :
- longueur est la longueur du chemin minimal entre id_r et id_s dans le sous-graphe ss_gph
- 0 < longueur < nbr_spt_sel

Appelé de : aff_s

Appelle :
lis_appart
lis_diff
gph_adj
lis_inter
lis_unio

Fonctionnement : on applique l'algorithme de Moore (version 2) pour calculer la distance entre le sommet id_r et le sommet id_s.

16. aff_creer_tab_aff

Objectif : creer la table aff pour permettre l'affichage des résultats.

Argument :
- r_c : liste par sous-graphe des remèdes centraux (variable globale à la méthode)
- ss_g : liste des sous_graphes (variable globale à la méthode)

Pre :
- r_c[i] contient la liste des remèdes centraux du sous_graphe contenu en ss_g[i]

Resultats :
- aff : table d'affichage des résultats (variable globale à la méthode)

Post :
- aff[i] contient une liste des sommets pour l'écran numéro i.

Appelé de : rc_calc

Appelle : /

Fonctionnement : pour chaque sous-graphe, on prend un à un les remèdes centraux. A chacun de ces remèdes centraux, on associe dans une liste (= un élément de aff) les différents symptômes faisant partie du même sous-graphe.

17. **aff_rc_display**

Objectif : affichage de l'écran numéro no_ecr

Argument : - no_ecr : numéro d'écran à afficher
- aff : table des écrans (variable globale à la méthode)

Pre : - 0 <= no_ecr < DIM_AFF
- aff contient les écrans du graphe dont CONNEC est la matrice de connection

Resultats : - affichage de l'écran no_ecr

Post : - l'écran affiché est celui contenu en aff[no_ecr]

Appelé de : rc_calc
rc_ecr_prec
rc_ecr_suiv

Appelle : fonctions d'écran
aff_r
aff_s

Fonctionnement : on affiche tout d'abord le titre,le nombre de remèdes centraux et le nombre de symptômes isolés puis on balaie la liste aff[no_ecr] pour affficher chaque sommet un à un. On affiche un sommet de type remède via aff_r et un sommet de type symptôme via aff_s.

18. **aff_r**

Objectif : affichage sur la ligne lig en première colonne du nom du remède correspondant au code remède id_r

Argument : - id_r : code remède
- lig : numéro de ligne

Pre : - 0 <= id_r < NBR_MED
- 3 <= lig <= 22

Resultats : - affichage du nom du remède

Post : - le nom du remède affiché sur la ligne lig en colonne 1 correspond au code remède id_r

Appelé de : aff_rc_display

Appelle : fonctions base de données
fonctions d'écran

Fonctionnement : on va chercher le nom du remède correspondant au code remède id_r et on l'affiche grâce aux fonctions d'écran.

19. **aff_s**

Objectif : affichage sur la ligne lig des données suivantes :

- col 26 : id_s
- col 53 : distance séparant le symptôme identifié par id_s et le remède identifié par id_r dans le sous-graphe gr

Argument :
- id_s : numéro du symptôme dans hst
- id_r : code remède
- lig : numéro de ligne
- gr : identificateur du sous-graphe

Pre :
- 0 <= id_s <= nbr_spt_sel
- 0 <= id_r < NBR_MED
- id_r identifie un remède qui est remède central dans le sous-graphe gr
- 3 <= lig <= 22

Resultats : - code du symptôme et distance affichés

Post :
- le code du symptôme id_s est affiché en colonne 26 sur la ligne lig
- la distance est la distance entre le symptôme de code id_s et le remède central de code id_r dans le graphe dont la matrice de connection est CONNEC. Cette distance est affichée sur la ligne lig en colonne 53.

Appelé de : aff_rc_display

Appelle :
fonctions base de données
fonctions d'écran
gph_lg_ch

Fonctionnement : on affiche le code remède sur la ligne lig en colonne 26 grâce aux fonctions d'écran puis on fait appel a la fonction gph_lg_ch pour calculer la distance entre le symptôme et le remède central. On affiche cette distance en colonne 53 grâce aux fonctions d'écran.

20. **lis_diff**

Objectif : la différence entre la liste p_1 et la liste p_2 est placée dans la liste p_res

Argument : - p_1 : liste de sommets
- p_2 : liste de sommets

Pre :

Resultats : - p_res : liste de sommets (variable globale de la méthode)

Post : - un sommet de p_res appartient à p_1 et n'appartient pas à p_2

Appelé de : gph_min_d
gph_ss_g
gph_lg_ch

Appelle : lis_appart

Fonctionnement : on balaie un à un les éléments de p_1. Si un élément n'appartient à p_2, on l'ajoute à p_res.

21. **lis_inter**

Objectif : intersection des listes p_1 et p_2 dans la liste p_res

Argument : - p_1 : liste de sommets
- p_2 : liste de sommets

Pre :

Resultats : - p_res : liste de sommets (variable globale de la méthode)

Post : - un sommet de p_res appartient à p_1 et à p_2

Appelé de : gph_min_d
gph_lg_ch

Appelle : lis_appart

Fonctionnement : on prend un à un les éléments de p_1 et on vérifie s'ils appartiennent à p_2. Si un élément de p_1 appartient à p_2, on l'ajoute à la liste résultat.

22. lis_unio

Objectif : union des listes p_1 et p_2 dans la liste p_res.

Argument : - p_1 : liste de sommets
- p_2 : liste de sommets

Pre :

Resultats : - p_res : liste de sommets (variable globale de la méthode)

Post : - un sommet de p_res appartient à p_1 ou à p_2

Appelé de : gph_min_d
gph_ss_g
gph_lg_ch

Appelle : lis_appart

Fonctionnement : on recopie dans p_res tous les éléments de p_1. On ajoute ensuite à p_res tous les éléments de p_2 qui ne font pas encore partie de p_res.

23. lis_egali

Objectif : vérification de l'égalité des listes p_1 et p_2

Argument : - p_1 : liste de sommets
- p_2 : liste de sommets

Pre :

Resultats : - 0 ou 1

Post : - 1 si les listes sont égales c'est-à-dire si tous les éléments de p_1 appartiennent à p_2 et si tous les éléments de p_2 appartiennent à p_1
- 0 sinon

Appelé de : gph_ss_g

Appelle : lis_appart

Fonctionnement : on vérifie que tout élément de p_1 appartient à p_2 et que tout élément de p_2 appartient à p_1. Il suffit qu'un seul élément d'une des deux listes n'appartiennent pas à l'autre liste pour qu'il y ait inégalité.

24. lis_appart

Objectif : vérification de l'appartenance de l'élément ident de type type à la liste p_lis.

Argument : - p_lis : liste de sommets
- ident : identificateur de l'élément
- type : type de l'élément

Pre :

Resultats : - present

Post : - present = 1 si l'élément appartient à la liste
- present = 0 sinon

Appelé de : gph_lg_ch
lis_inter
lis_unio
lis_diff
lis_egali

Appelle : /

Fonctionnement : on balaie la liste p_lis en vérifiant pour chaque élémént si on a en même temps que :

- identificateur de l'élément = ident
- type de l'élément = type.

25. **lis_tri**

Objectif : tri de la liste p_lis dans p_res en mettant en tête de liste les éléments de type S classés en ordre croissant sur ident suivis des éléments de type R.

Argument : - p_lis : liste de sommets

Pre : - p_lis contient au moins un sommet de type S

Resultats : - p_res : liste de sommets (variable globale de la méthode)

Post :
- tout sommet de p_lis appartient à p_res
- tout sommet de p_res appartient à p_lis
- pour tout sommet de type S :
  - s'il y a un sommet précédent, il est de type S et ident du précédent est inférieur à ident du sommet
  - s'il y a un sommet suivant, il est soit de type R soit de type S. S'il est de type S alors ident du suivant est supérieur à ident du sommet

Appelé de : gph_ss_g

Appelle :

Fonctionnement :

26. lis_ajout

Objectif : ajout d'un élément en fin de liste

Arguments :
- ident_el : identificateur d'élément
- type_el : type d'élément
- p_der (variable globale de la méthode)

Pre :

Resultats :
- p_der.type
- p_der.ident
- p_der

Post :
- p_der.type contient type_el
- p_der.ident contient ident_el
- p_der pointe vers une nouvelle place où ajouter

Appelé de :
aff_creer_tab_aff
gph_rc_ss_g
gph_ss_g
gph_sts
gph_sts_isol
gph_adj
lis_inter
lis_unio
lis_diff
lis_tri

Appelle : /

Fonctionnement : il suffit simplement de recopier et d'allouer une nouvelle place pour p_der.

28. fonctions BD

Les fonctions d'accès à la base de données sont déjà définies dans le logiciel RADAR [BD]. Elle servent aux autres méthodes implémentées dans le logiciel. Les fonctions qui nous sont utiles dans l'implémentation de la méthode du remède central sont essentiellement :

- l'accès à la liste des codes des remèdes d'un symptôme
- l'accès au nom d'un remède en fonction de son code.

29. fonctions d'écran

Le logiciel RADAR utilise un certain nombre de fonction pour afficher des données à l'écran. L'écran à afficher peut être bufferisé et affiché d'un seul coup quand on le demande. Les fonctions les plus importantes que nous utilisons sont les suivantes :

- positionnement à un endroit de l'écran
- affichage d'une donnée (numérique, caractère , ...) à l'endroit où on est positionné
- mise à blancs de l'écran ou d'une partie de l'écran.

CHAPITRE VI : Resultats et constatations.

Les différents ensembles de symptômes qui nous servent de bases pour calculer les résultats sont des cas qui se sont effectivement présentés aux médecins homéopathes. Ces cas cliniques ont été guéris grâce à la méthode Vithoulkas. Le remède prescrit était chaque fois le premier remède suggéré par cette méthode.

Ces cas font appel non seulement aux listes de remèdes du répertoire de Kent mais également, pour certains symptômes de certains cas, aux listes de remèdes d'un autre répertoire : le Synthesis. Ce répertoire est organisé de la même façon que le répertoire de Kent. L'accès se fait donc de la même manière. Il est toutefois plus actuel que le répertoire de Kent. Certains médecins homéopathes y font beaucoup appel pour certaines catégories de symptômes.

VI.1. Cas cliniques.

Nous présentons, à partir de la page suivante 6 cas, cliniques un par un. Pour chaque symptôme, nous indiquons :

- en première colonne, s'il a été choisi dans le répertoire de Kent ou dans le Synthesis
- en deuxième colonne, le poids attribué par le médecin à ce symptôme
- en troisième colonne, le numéro attribué à ce symptôme suivi de l'intitulé du symptôme.

VI.1.1. Cas numéro 1.

S 2 1- MIND - ANXIETY - conscience, as if guilty of a crime
S 2 2- MIND -SUICIDAL disposition -knife -with
S 2 3- MIND -IRRESOLUTION, indecision
S 3 4- MIND - CONFIDENCE, want of self
S 2 5- MIND - DREAM, asif in a
S 3 6- MIND - CONFUSION of mind
K 3 7- MOUTH - TASTE, - metallic
S 2 8- MIND - MEMORY,WEAKNESS OF
K 3 9- MOUTH - ODOR (breath) - offensive
S 3 10- GENERAL - PERSPIRATION, -after p. - agg.
S 1 11- MIND - WEEPING,tearful mood
S 2 12- MIND -CONSOLATION,kind words - agg.
S 2 13- GENERAL - WARM - agg.
K 3 14- STOMACH - THIRST -extreme

VI.1.2. Cas numéro 2.

K 3 1- HEAD - PAIN,headache in general - reading - agg.
K 3 2- HEAD - PAIN,headache in general - writing,from
K 3 3- HEAD - PAIN,headache in general - pressing (See Bursting,Drawing) - Temples
K 3 4- HEAD - PAIN,headache in general - pressing (See Bursting,Drawing) - Temples - reading
S 2 5- MIND - CONFUSION of mind - morning - waking,on
S 2 6- MIND - FEAR,apprehension,dread - crowd,in a
S 2 7- GENERAL - COLD in general agg.
K 2 8- HEAD - PAIN,headache in general - music,from
K 2 9- HEAD - PAIN,headache in general - noise,from
S 1 10- MIND - COWARDISE
S 1 11- MIND - DISCOURAGED
S 1 12- MIND - FEAR,apprehension,dread - death,of
S 1 13- GENERAL - AFTERNOON(13-18h) - 16h
S 1 14- GENERAL - WEATHER - wet w. - agg.
K 1 15- HEAD - PAIN,headache in general - menses - before
K 1 16- HEAD - PAIN,headache in general - Temples -right - to left
K 1 17- HEAD - PAIN,headache in general - Pressing (See Bursting,Drawing) - Temples - right - to left
K 1 18- STOMACH - NAUSEA - fasting,while
K 1 19- STOMACH - NAUSEA - menses, -during
K 1 20- STOMACH - NAUSEA - riding in a carriage or on the ears,, while
K 1 21- STOMACH - PAIN - eating, - after
S 1 22- MIND - FEAR,apprehension,dread - alone,of being
S 1 23- MIND - FEAR,apprehension,dread - people,of; anthropophobia
S 2 24- MIND - FORGETFUL
S 1 25- MIND - IRRITABILITY - WAKING,on

VI.1.3. Cas numéro 3.

K 3 1- HEAD - PAIN,headache in general - reading - agg.
K 3 2- HEAD - PAIN,headache in general - writing,from
K 3 3- HEAD - PAIN,headache in general - pressing (See Bursting,Drawing) - Temples
K 3 4- HEAD - PAIN,headache in general - pressing (See Bursting,Drawing) - Temples - reading
S 2 5- MIND - CONFUSION of mind - morning - waking,on

VI.1.4. Cas numéro 4.

S 3 1- MIND DISCONTENTED,displeased,dissatisfied
S 3 2- MIND IRRITABILITY
S 3 3- GENERAL COLD in general agg.
S 2 4- MIND CONFIDENCE, want of self
S 2 5- MIND DULLNESS,sluggishness.,diff.of thinking, comprehension
S 2 6- MIND FORGETFUL
S 2 7- MIND IRRESOLUTION,indecision
S 2 8- MIND IRRITABILITY,morning
S 2 9- MIND DULLNESS,mental exertion,from
K 3 10- FACE PAIN(aching,prosopalgia) left
K 3 11- FACE PAIN(aching,prosopalgia) forenoon 9 a.m.
K 1 12- FACE PAIN(aching,prosopalgia) cold air agg.
K 2 13- FACE PAIN(aching,prosopalgia) draft agg.
K 2 14- FACE PAIN(aching,prosopalgia) lying,while
K 1 15- FACE PAIN(aching,prosopalgia) afternoon
S 1 16- MIND ANXIETY fear,with

VI.1.5. Cas numéro 5.

K 1 1- EYE - LACHRYMATION (See tears) - Looking steadily
K 1 2- LARYNX AND TRACHEA. - VOICE - hoarseness
K 1 3- EXTREMITIES. - TENSION - Thigh
K 1 4- NOSE - DISCHARGE, - watery
K 1 5- COUGH. - MOTION - agg.

VI.1.6. Cas numéro 6.

K 3 1- ABDOMEN - PAIN,aching,dull pain (See Boring,Drawing, Distress,Digging,Gnawing,Pressing,etc.) - bed,in
K 3 2- ABDOMEN - PAIN,aching,dull pain (See Boring,Drawing, Distress,Digging,Gnawing,Pressing,etc.) - cold, becoming from
K 3 3- ABDOMEN - PAIN,aching,dull pain (See Boring,Drawing, Distress,Digging,Gnawing,Pressing,etc.) - cramping, griping - stool, - after - amel.
K 2 4- MIND - COMPANY,DESIRE for;aversion to solitude, company am.

VI.2. Résultats.

Pour chaque cas, nous précisons les résultats obtenus avec chacune des trois méthodes : répertorisation classique, méthode Vithoulkas et méthode du remède central. Pour la répertorisation classique, nous reproduisons le tableau obtenu en limitant ce tableau aux huit premiers remèdes. Nous indiquons quel est le premier remède suggéré par la métode Vithoulkas. C'est donc le remède qui a guéri le cas. En ce qui concerne la méthode du remède central, nous donnons la liste des remèdes centraux lorsque celle-ci n'est pas trop longue.

VI.2.1. Cas numéro 1.

- Répertorisation classique :

	1	1	1	4	4	4	4	8
	carb-v.	merc.	nux-v.	bry.	kali-bi.	nat-m.	seneg.	chin.
	4/9	4/9	4/9	4/8	4/8	4/8	4/8	4/7
	4/9	4/9	4/9	4/8	4/8	4/8	4/8	4/7
1-	-	-	-	-	-	-	2	-
2-	3	3	2	3	3	3	2	2
3-	2	2	2	1	2	2	-	1
4-	2	3	3	2	2	2	2	2
5-	2	1	2	2	1	1	2	2

- Méthode Vithoulkas :

osm.

- Remède central : il y a 97 remèdes centraux.

VI.2.3. Cas numéro 3.

- Répertorisation classique :

1	2	3	4	5	6	7	8
nat-m.	calc.	lyc.	lach.	sil.	carb-v.	glon.	plat.
5/13	4/7	4/7	3/7	4/6	3/6	3/5	2/5
14/38	11/20	11/19	8/18	11/16	8/16	9/15	6/15

1-	3	2	1	1	1	1	1	2
2-	3	2	1	-	2	-	1	-
3-	3	2	3	3	1	3	3	3
4-	3	-	-	-	-	-	-	-
5-	1	1	2	3	2	2	-	-

- Méthode Vithoulkas :

nat-m.

- Remède central : il y a un seul remède central

nat-m.

VI.2.4. Cas numéro 4.

- Répertorisation classique :

1	2	3	4	5	6	7	8
lyc.	nat-m.	calc.	phos.	ars.	nux-v.	puls.	lach.
22/50	17/32	19/37	17/31	17/31	18/29	16/30	13/26
33/71	29/63	30/58	27/48	26/45	26/41	22/41	21/41

1-	1	3	2	-	-	1	-	1
2-	1	3	2	1	1	-	-	-
3-	3	3	2	2	2	1	2	3
4-	-	3	-	-	-	-	-	-
5-	2	1	1	2	1	-	1	3
6-	2	2	1	1	1	2	2	-
7-	3	2	3	3	3	3	2	1
8-	-	-	-	2	-	1	-	-
9-	1	-	3	2	2	2	-	2
10-	3	1	1	1	-	1	2	-
11-	2	1	2	1	2	1	2	2
12-	2	2	3	3	3	2	2	2
13-	3	1	-	-	1	2	1	-
14-	2	-	3	1	3	1	3	2
15-	2	2	2	1	1	1	2	2
16-	-	-	-	-	-	-	-	-
17-	1	-	-	-	-	-	-	-
18-	3	-	2	-	-	-	-	-
19-	2	1	2	1	1	3	2	-
20-	2	-	2	-	6	2	-	-
21-	3	2	2	3	3	3	3	3
22-	3	-	1	3	3	1	2	-
23-	3	2	1	1	1	-	2	1
24-	3	2	2	3	1	1	1	2
25-	3	1	-	-	2	1	1	2

- Méthode Vithoulkas :

nat-m.

- Remède central : il y a dix remèdes centraux.

caust.	lach.	lyc.	nux-v.
sulph.	ars.	bell.	carb-an.
kali-p.	puls.		

VI.2.5. Cas numéro 5.

- Répertorisation classique :

1	2	3	4	5	6	7	8
merc.	nat-m.	sulph.	calc.	ars.	sil.	sep.	lach.
14/31	13/30	13/29	14/28	12/26	13/24	11/25	11/23
33/74	31/70	31/68	33/66	28/61	31/59	26/59	26/55

1-	2	2	3	1	3	2	-	2
2-	2	-	-	2	2	-	-	-
3-	2	2	2	2	2	2	2	3
4-	1	2	1	1	-	3	-	1
5-	1	2	2	2	1	1	1	2
6-	3	3	2	3	2	3	3	3
7-	3	1	2	2	2	1	2	2
8-	3	2	2	2	3	2	3	3
9-	3	3	3	2	3	1	2	3
10-	2	1	2	2	1	1	3	-
11-	3	3	3	3	2	1	3	1
12-	1	3	1	1	2	3	3	-
13-	2	3	3	2	-	1	1	2
14-	3	3	3	3	3	3	2	1

- Méthode Vithoulkas :

merc.

- Remède central : il y a deux remèdes centraux.

calc. et merc.

VI.2.6. Cas numéro 6.

- Répertorisation classique :

	1	2	3	4	5	6	7	8
	nux-v.	ars.	phos.	coloc.	gels.	mag-c.	kali-c.	nat-s.
	4/11	2/6	2/5	2/4	2/4	2/3	2/4	1/3
	11/31	5/15	5/12	5/11	5/10	6/9	5/9	3/9
1-	3	-	-	-	-	1	1	-
2-	3	3	2	-	-	-	-	-
3-	3	-	-	3	2	2	-	3
4-	2	3	3	1	2	-	3	-

- Méthode Vithoulkas :

nux-v.

- Remède central : il y a un seul remède central.

nux-v.

VI.3. Constatations.

VI.3.1. Le nombre de remèdes centraux.

Malgré le nombre réduit de cas dont nous disposons, nous pouvons tout de même faire quelques constatations sur les résultats obtenus.

Les résultats obtenus par la méthode du remède central pour les cas 3, 5 et 6 sont assez encourageants. Chaque fois, la méthode suggère un très petit nombre de remèdes (1 ou 2). Le remède effectivement prescrit correspond au remède suggéré (dans le cas d'un seul remède central) ou bien fait partie de la liste des remèdes suggérés (dans le cas de 2 remèdes centraux).

Etait-il toutefois nécessaire d'appliquer la méthode du remède central pour obtenir ce résultat ? Le tableau obtenu par la méthode classique de répertorisation laissait déjà apparaître ce résultat. Ce tableau peut, en effet, être considéré comme une partie de la matrice de connection du graphe. Prenons par exemple le tableau de répertorisation classique du cas numéro 3. Nous constatons que le remède nat-m. se trouve en correspondance directe, ou plutôt à distance 1 de tous les symptômes sélectionnés. Or il est le seul à présenter cette caractéristique. Il est donc le seul remède pour lequel on ait min(min dist R) = 1. Ce qui veut dire qu'il est remède central unique pour ce cas.

La même chose se produit pour le cas numéro 5. Ici, deux remèdes (calc. et merc.) ont une correspondance directe avec tous le symptômes. Ces deux remèdes sont centraux puisqu'ils sont les seuls pour lesquel on ait min(min dist R) = 1.

Dans ces cas-là, où certains remèdes se trouvent en regard de tous les symptômes, il semble que le calcul du remède central ne soit pas nécessaire. Une simple déduction sur le tableau de la répertorisation classique suffit.

Les résultats du cas numéro 4 par la méthode du remède central est encore supportable - le médecin se voit suggéré une liste de 10 symptômes parmi lesquel il peut choisir - même si on peut s'étonner de ne pas voir dans cette liste des remèdes, le remède effectivement prescrit (nat-m.), obtenu par la méthode Vithoulkas. Cela ne veut absolument pas dire que ces dix remèdes ne seraient pas efficaces pour ce cas. Il faudrait pour le vérifier prendre le risque d'essayer l'un ou

l'autre de ces remèdes sur un patient présentant tous ces symptômes. D'après la méthode , les dix remèdes sont sur un strict pied d'égalité. Ils sont aussi valables les uns que les autres.

Les résultats des cas numéro 1 et numéro 2 sont tout simplement catastrophiques. Comment, en effet, choisir un remède à prescrire dans une liste de 67 ou 97 remèdes qui sont tous aussi valables les uns que les autres puisque tous sont centraux.

VI.3.2. Le temps de réponse.

La méthode, programmée en langage C, tourne sur un OLIVETTI AT&T 3B2/400 sous Unix connecté au PACX. Les résultats obtenus l'ont été à un moment où nous étions seul occupés à travailler sur l'Olivetti. Malgré cela, le temps de réponse calculé entre le moment où nous demandons le calcul des résultats (après avoir sélectionné les symptômes) et le moment où apparaissent les résultats est assez catastrophique pour la méthode du remède central. Voici approximativement ces temps de réponse pour chaque cas, pour chacune des méthode :

a) Cas numéro 1 :

- répertorisation classique : 3 secondes
- méthode Vithoulkas : 10 secondes
- remède central : 2 minutes 20 sec.

b) Cas numéro 2 :

- répertorisation classique : 7 secondes
- méthode Vithoulkas : 31 secondes
- remède central : 32 minutes

c) Cas numéro 3 :

- répertorisation classique : 4 secondes
- méthode Vithoulkas : 1 minutes 34 sec.
- remède central : 46 secondes

d) Cas numéro 4.

- répertorisation classique : 5 secondes
- méthode Vithoulkas : 41 secondes
- remède central : 24 minutes

e) Cas numéro 5.

- répertorisation classique : 5 secondes
- méthode Vithoulkas : 29 secondes
- remède central : 28 minutes

f) Cas numéro 6.

- répertorisation classique : 3 secondes
- méthode Vithoulkas : 5 secondes
- remède central : 8 secondes

Les seuls temps raisonnables pour la méthode du remède central sont ceux du cas numéro 6 et du cas numéro 3. La méthode donne pour ces cas le même remède que la méthode Vithoulkas en 8 et 46 secondes environ. Cette "rapidité" de réponse vient du fait que les listes de remèdes des symptômes sont assez courtes. La recherche est donc moins lourde puisqu'on vérifie pour un nombre réduit de remède s'ils sont remèdes centraux ou non. Toutefois, nous l'avons déjà dit, un rapide coup d'oeil au tableau obtenu par la répertorisation classique nous aurait permis d'arriver au même résultat certainement aussi vite.

Pour le cas numéro 1, nous obtenons des résultats après 2 minutes 20 secondes environ. Ce n'est pas catastrophique. Mais la réponse, elle, l'est, puisque nous avons vu que pour ce cas, la méthode du remède central suggère 97 remèdes centraux.

Les temps de réponse des cas numéro 2, numéro 4 et numéro 5 sont évidemment excessifs. Il serait étonnant de voir un médecin attendre environ 1/2 heure un résultat devant son écran sachant que ce résultat sera peut-être une liste de plusieurs dizaines de remèdes (comme dans le cas numéro 2). La méthode proposera peut-être une liste plus réduite (comme dans le cas numéro 5) mais cette liste comme nous l'avons vu

peut être déduite du tableau obtenu par la méthode classique de répertorisation beaucoup plus rapidement.

VI.4. Améliorations.

VI.4.1. Diminution du nombre de remèdes centraux.

La méthode implémentée telle qu'elle est exposée au chapitre III ne permet pas dans certains cas d'apporter une aide au médecin homéopathe. La liste des remèdes proposée est parfois beaucoup trop longue. Il semble que si on n'a pas un ou quelques remèdes centraux qui apparaissent immédiatement (par la méthode de répertorisation classique), on doive s'attendre à une liste de remèdes centraux assez longue. Les listes de remèdes en regard des symptômes sélectionnés sont, souvent très longues (parfois plus de 300 remèdes) et on obtient un graphe avec énormément de connections dans lequel on ne sait pas dégager un petit nombre de remèdes centraux.

L'idéal serait de sélectionner des symptômes présentant moins de remèdes, c'est-à-dire plus précis.

Une autre solution pour réduire la liste des remèdes suggérés pourrait être d'affiner la méthode en tenant compte par exemple des poids attribués aux symptômes par le médecin. La méthode du remède central est la seule à ne pas tenir compte de ces poids. C'est peut-être une erreur. On pourrait par exemple pondérer les arêtes du graphe et tenir compte de cette pondération dans le calcul de la distance minimale.

VI.4.2. Diminution du temps de réponse.

Le temps de réponse est évidemment trop grand dans certains cas. Il est dû au trop grand nombre de remèdes du graphe, pour chacun desquels il faut vérifier s'il est remède central. Un choix de symptômes plus précis (présentant moins de remèdes) serait une solution.

On pourrait certainement penser à tenter d'optimiser le programme. Aucun effort n'a été fait dans ce sens. Au niveau algorithmique, il devrait être possible de tenir compte du fait que nous avons un graphe tout-à-fait particulier composé de sommets de deux types et d'arêtes permettant le passage dans les deux sens entre deux sommets.

CONCLUSION.

Nous sommes obligé de terminer ce travail sur un constat de semi-échec. La méthode que nous avons implémentée ne nous a pas fourni les résultats que nous espérions. Notre implémentation et les résultats qu'elle nous a donnés nous a toutefois permis de soulever quelques problèmes de la méthode du remède central. Le nombre limité de cas dont nous disposions ne nous permet pas de tirer des conclusions mais les problèmes rencontrés risquent de surgir pour d'autres cas que ceux que nous avons testés.

Nous suggérons, comme nous l'avons dit dans le chapitre VI d'affiner la méthode, notamment en tenant compte des poids des symptômes, et de voir les résultats ainsi obtenus. Si ces résultats s'avèrent intéressants, nous pourrions envisager de chercher à optimiser le programme par de nouveau algorithmes ou par des modifications dans la programmation.

Nous pourrions envisager également d'implémenter les autres fonctions décrites au chapitre IV, fonctions pour lesquelles une analyse des besoins réels des homéopathes serait nécessaire. Le traitement des dossiers patient est certainement une chose intéressante qui mérite une analyse plus approfondie.

BIBLIOGRAPHIE.

[ALLEN] T. F. A. Allen

The encyclopedia of Pure Materia Medica (12 vol.). B. Jain Publishers, New Delhi.

[GRAPHE] J. Fichefet

Théorie des graphes et son algorithmique. Cours de première licence en informatique. FNDP, Namur.

[RADAR] J. Fichefet, A. Jacques, P. Gardin, J. Paris

RADAR : a knowledge-based expert system in homeopathy.

[HAHN] S. F. Hahnemann

Exposition de la doctrine médicale homéopathique, ou Organon de l'art de guérir. Traduit de l'allemand. Paris, J.-B. Baillière, Librairie de l'académie royale de médecine.

[KENT] J. T. Kent

Repertory of the Homeopathic Materia Medica, Ehrart and Karl, Chicago.

[LANG-C] B. W. Kernighan, D. M. Ritchie

Le langage C. Masson, 1984.

[BD] Mode d'emploi des fonctions d'accès à la base de données.

[MANUAL] RADAR. User's manual.

[RESCONI] G. Resconi, M. Trionfi

Relation on the systemic homeopathy.

[MDL] A. Van Lamsweerde

Méthodologie de développement de logiciel. Cours de deuxième licence en informatique. FNDP, Namur.

ANNEXES : Programmes de la méthode

```
/*****************************************************/
/* fichier de définition des variables globales      */
/*    à la méthode                                   */
/*****************************************************/



#include <stdio.h>

#define DIM_RC 50
#define DIM_AFF 50 /* nombre de page maximum pour l'affichage
                      des resultats */
#define CONNEC(i,j) *(connec[i]+j)

unsigned char *connec[DIM_RC+3];

struct elist { int ident;
        unsigned char type;
        struct elist *next;

        }   *p_der,*p_res,*l_rc,*ss_g[DIM_RC],*r_c[DIM_RC],
            *aff[DIM_AFF] ,*isol_s;

short nbr_ss_g;
_bool rc_done;
short nbr_rc;
short nbr_spt_isol;
```

```
£include "/usr/archimed/navi/radardef.c"
£include "rcdef.c"


/* routines  rc()
             gph_connec_fill()
             rc_calc()
             aff_rc_display(no_ecr)
             aff_r(id_r,lig)
             aff_s(id_s,lig)
             rc_ecr_prec(pno_ecr)
             rc_ecr_suiv(pno_ecr)
             rc_del()                              to be done
             aff_spt_isol()                        to be done
             aff_creer_tab_aff()


             gph_rc_all()
             gph_rc_ss_g(ss_gph)
             gph_min_d(id_r,ss_gph)
             gph_ss_g()
             gph_lg_ch(id_r,id_s,ss_gph)
             gph_sts()
             gph_simpl()
             gph_spt_isol()
             gph_adj(p_lis)
             gph_an_co(j)
             gph_an_li(i)


            lis_inter(p_1,p_2)
            lis_union(p_1,p_2)
            lis_diff(p_1,p_2)
            lis_egali(p_1,p_2)
            lis_ajout(ident_el,type_el)
            lis_appart(p_lis,ident,type)
            lis_tri(p_lis)


            aff_mat()                          to be deleted
            aff_lis(p)                         to be deleted


*/
```

```
rc()

{
short no_ecr;
int i;

no_ecr = 0;
rc_done = 0;

INIT :

_clear(_SCREEN,_ALL);

while (1)
  { B_lit_commande("enter command,please >> ",'a');
    switch(_commande[0])

      {case _ESC :
          if (!rc_done) {_error(_message("00"),2);
          break;}

          switch(_commande[2])

           {case 'E' :
            case 'F' : rc_ecr_prec(&no_ecr);
                       break;

            case 'G' :
            case 'H' : rc_ecr_suiv(&no_ecr);
                       break;

           }
        break;

       case 'u' : rc_ecr_prec(&no_ecr);
                  break;

       case 'd' : rc_ecr_suiv(&no_ecr);
                  break;

       case 'c' : if (nbr_spt_sel < 2)
                    {_error("Not enough symptoms selected",3);
                     B_lit_commande("return to continue",'a'); }

                  else
                  rc_calc();
                  break;

       case 'q' : return;

       case 's' : do s_display();
                  while (hst_ctrl() < 0);
            if ((nbr_spt_sel == 0) || (_commande[1] == '!'))
            {
```

```
                if (rc_done) { for (i=0;i<DIM_RC;i++)
                                   free(connec[i]);
                                   rc_done = 0;
                                    }
    return;
    }
    goto INIT;

       case 'a' : if (!rc_done)
                  {_error(_message("00"),2);
                   break;}
                  rc_del();
                  /* ....  */
                  break;

       case 'i' : if (!rc_done)
                  {_error(_message("00"),2);
                   break;}
                  aff_spt_isol();
                  break;

       default : _error(_message("00"),3);

      } /* end of switch */

    } /* end of while */

} /* end of rc */

/* ------------------------------------------------------------ */
```

```
gph_connec_fill()

/* ------------------------------------------------------------*/
/* cette fonction remplit la matrice CONNEC a partir des      */
/* symptomes de hst                                           */
/* ------------------------------------------------------------ */

{ int k,l,i_1;


for (k = 0;k< DIM_RC+3;k++)
   connec[k] = (unsigned char *) malloc(NBR_MED+2);

for (k = 0;k<DIM_RC+3;k++)
  for (l = 0;l<NBR_MED+2;l++)
     CONNEC(k,l) = 0;


for (i_1 = 0;i_1 < nbr_spt_sel;i_1++)

  {_adj(LST_MED,_DIR,_RCD,hst[i_1].adr_rmd,_INP);
   k = _IDX(LST_MED,_RCD);
   l = hst[i_1].lgr_rmd;

   while (l > 0)

     {CONNEC(i_1,lst_med[k]) = 1;
      l--;
      k = _idx_inc(LST_MED,_RCD,_INP);
     }

  } /* end for */

aff_mat();

} /* end gph_connec_fill */


/* ------------------------------------------------------------ */
```

```
rc_calc()

/* ------------------------------------------------------------ */
/* fonction de calcul et d'affichage du ou des remede(s)        */
/* central(aux)                                                 */
/*                                                              */
/* nbr_spt_sel >= 2                                             */
/* ------------------------------------------------------------ */

{

gph_connec_fill();
gph_simpl();
gph_rc_all();

gph_spt_isol();

if (nbr_rc != 0)
{aff_creer_tab_aff();
aff_rc_display(0);}
else _error("no treatment centre but only isolated
symptoms",2);
rc_done = 1;

} /* end rc_calc */

/* ------------------------------------------------------------ */
```

```
aff_rc_display(no_ecr)

/* ------------------------------------------------------------ */
/* affichage de l'ecran no_ecr                                  */
/*                                                              */
/* 0 <= no_ecr <DIM_AFF                                         */
/* aff[no_ecr] est non vide                                     */
/* ------------------------------------------------------------ */

short no_ecr;

{struct elist *p,*p_ss_g;

 int k;                 /* indice ligne */
 int r_cou;             /* identificateur du remede courrant */
 int fin;

k = 3;
p_ss_g = aff[no_ecr];
p = p_ss_g->next;

B_locate(top_lin -1,1);
B_clear(_SCREEN,_RGT);

_locate(24,1);printf("%d treatment centre(s)",nbr_rc);
_locate(24,35);printf("%d isolated symptom(s)",nbr_spt_isol);

B_sprintf("TREATMENT CENTRE(S)");
B_locate(top_lin-1,26);B_sprintf("SYMPTOM");
B_locate(top_lin-1,53);B_sprintf("AT DISTANCE");
fin = 0;

while(fin == 0)

  {if (p->type = 'R')
    {aff_r(p->ident,k);
     r_cou = p->ident;}
   else {aff_s(p->ident,r_cou,k,p_ss_g->ident);
  k++;}
   if (p->next != NULL)
     {p_ss_g = p->next;
      p = p_ss_g->next;}
   else fin = 1;
  }

} /* end aff_rc_display */

/*------------------------------------------------------------ */
```

```
aff_r(id_r,lig)

/*------------------------------------------------------------ */
/* affichage du remede id_r sur la ligne lig                   */
/*------------------------------------------------------------ */

int id_r;
int lig;

{

 _adj(NOM_MED,_DIR,_RCD,id_r,_INP);
 id_r = _IDX(LST_MED,_RCD);
 B_locate(lig,1);
 B_sprintf(nom_med[id_r]);

} /* end aff_r */

/*------------------------------------------------------------ */



aff_s(id_s,id_r,lig,gr)

/*------------------------------------------------------------ */
/* affichage sur la ligne lig du numero de symptome id_s       */
/* et de la distance entre le symptome id_s et le remede       */
/* id_r dans le sous-graphe ss_g[gr]                           */
/*------------------------------------------------------------ */

int id_s;
int id_r;
int lig;
int gr;

{

B_locate(lig,26);
B_nprintf(id_s);
B_locate(lig,53);
B_nprintf(gph_lg_ch(id_r,id_s,ss_g[gr]));

} /* end aff_s */

/*------------------------------------------------------------ */
```

```
rc_ecr_prec(pno_ecr)

/* ------------------------------------------------------------ */
/* fonction d'affichage de l'ecran precedent des resultats */
/* ------------------------------------------------------------ */

short *pno_ecr;

{
if (!rc_done)
_error(_message("00"),2);
else if (*pno_ecr > 0)
     {*pno_ecr--;
      aff_rc_display(*pno_ecr);}
     else _error("NO PREVIOUS SCREEN",2);

} /* end rc_ecr_prec */

/*------------------------------------------------------------ */



rc_ecr_suiv(pno_ecr)

/* ------------------------------------------------------------ */
/* fonction d'affichage de l'ecran suivant des resultats   */
/* ------------------------------------------------------------ */

short *pno_ecr;

{
if (!rc_done)
_error(_message("00"),2);
else if (aff[*pno_ecr+1] != NULL)
     {*pno_ecr++;
      aff_rc_display(*pno_ecr);}
     else _error("NO MORE SCREEN",2);

} /* end rc_ecr_suiv */

/*------------------------------------------------------------ */
```

```
rc_del()

/* ------------------------------------------------------------ */
/* fonction de suppression d'un ou plusieurs RC                 */
/* ------------------------------------------------------------ */

{
}

/*------------------------------------------------------------- */


aff_spt_isol()

/* ------------------------------------------------------------ */
/* fonction d'affichage des symptomes isoles                    */
/* ------------------------------------------------------------ */

{
}

/*------------------------------------------------------------- */
```

```
aff_creer_tab_aff()

/* ------------------------------------------------------------ */
/* fonction de creation de la table d'affichage aff             */
/*                                                              */
/* un element de cette table point vers une liste de S et       */
/* R a afficher sur un ecran                                    */
/* ------------------------------------------------------------ */

{ struct elist *p_interm_1,*p_interm_2,*memp;
  int cpt;
  int i; /* indice dans la table r_c */
  int j; /* indice dans la table aff */

for (j = 0;j < DIM_AFF;j++)
  aff[j] = NULL;

i = 0;
j = 0;
cpt = 0;
p_der = (struct elist *) malloc(sizeof(*p_der));
aff[j] = p_der;

while (i < nbr_ss_g)

  {p_interm_1 = r_c[i];
   while (p_interm_1 != NULL)

      {lis_ajout(i,'G');
       lis_ajout(p_interm_1->ident,'R');
       p_interm_2 = ss_g[i];
       while ((p_interm_2 ->type != 'R') && (p_interm_2 !=
       NULL))

 {lis_ajout(i,'G');
  memp = p_der;
  lis_ajout(p_interm_2 ->ident,'S');
  cpt++;
  if (cpt == 20)
  {cpt = 0;
   memp = NULL;
   j++;
   aff[j] = p_der;
   if (p_interm_2->type = 'S')
   lis_ajout(i,'G');
   lis_ajout(p_interm_1->ident,'R');
         }
  p_interm_2 = p_interm_2->next;

 } /* end while p_interm_2 */
 p_interm_1 = p_interm_1->next;

       } /* end while p_interm_1 */
       i++;
```

```
        }/* end while (i < nbr_ss_g) */
} /* end aff_creer_tab_aff */
/*------------------------------------------------------------------ */
```

```
gph_rc_all()

/*------------------------------------------------------------ */
/* cette fonction renvoie dans le tableau r_c[DIM_RC] la      */
/* liste des RC pour chaque ss-graphe de ss_g[DIM_RC]         */
/*                                                            */
/* elle met a jour nbr_rc                                     */
/*------------------------------------------------------------ */

{ int k;

k= 0;
nbr_rc = 0;

gph_ss_g();  /* distinction des sous_graphe  */


k = 0;
while (k <= nbr_ss_g - 1)

  /* pour tout sous-graphe, calculer le(s) RC  */

  { if (ss_g[k]->next == NULL)
    r_c[k] = NULL;
    else
    {gph_rc_ss_g(ss_g[k]);
    r_c[k] = l_rc;}
    k++;
  }

} /* end gph_rc_all */

/*------------------------------------------------------------ */
```

```
gph_rc_ss_g(ss_gph)

/* ------------------------------------------------------------ */
/* cette fonction calcule le(s) RC du sous-graphe ss_gph        */
/* elle renvoie dans l_rc la liste du(des) RC du                */
/* sous-graphe ss_gph                                           */
/* ss_gph doit contenir plus d'un sommet                        */
/* ------------------------------------------------------------ */

struct elist * ss_gph;

{ struct elist *p,*memp,*p_interm;
  int min,k;

min = DIM_RC * 2 -2;

while (p != NULL)

  {
     if (p->type == 'R')

     {k = gph_min_d(p->ident,ss_gph);
      if (k < min)
      min = k;
     }
    p = p->next;

  }

l_rc = (struct elist *) malloc(sizeof(*l_rc));
p_interm = l_rc;
p = ss_gph;

while (p != NULL)

  {if (p->type == 'R')

   {k = gph_min_d(p->ident,ss_gph);
    if (k == min)

     {nbr_rc++;
      memp = p_interm;
      p_der = p_interm;
      lis_ajout(p->ident,'R');
      p_interm = p_der;
     }
   }

    p = p->next;
  }
memp->next = NULL;
} /* end gph_rc_ss_g */

/*------------------------------------------------------------ */
```

```
gph_min_d(id_r,ss_gph)

/*------------------------------------------------------------------ */
/* cette fonction calcule la distance minimun pour le               */
/* sommet id_r du sous-graphe ss_gph                                */
/* id_r est de type 'R'                                             */
/*------------------------------------------------------------------ */

int id_r;
struct elist *ss_gph;

{ struct elist *p_x,*p_s,*p_d,*p_interm_1,*p_interm_2;
  int k;


p_x = ss_gph;
k = 0;

p_d = (struct elist *) malloc(sizeof(*p_d));
p_d->ident = id_r;
p_d->type = 'R';
p_d->next = NULL;
p_s = (struct elist *) malloc(sizeof(*p_s));
p_s->ident = id_r;
p_s->type = 'R';
p_s->next = NULL;


while (p_s != NULL)

  { lis_diff(p_x,p_d);
    p_interm_2 = p_res;
    gph_adj(p_s);
    p_interm_1 = p_res;
    lis_inter(p_interm_1,p_interm_2);
    p_s = p_res;
    lis_unio(p_d,p_s);
    p_d = p_res;
    k++;
  }

return(k-1);

} /* end gph_min_d */

/*------------------------------------------------------------------ */
```

```
gph_ss_g()

/*-------------------------------------------------------------*/
/* cette fonction distingue les differents sous-graphes        */
/* elle range dans le tableau ss_g les listes des sommets      */
/* des differents sous_graphes                                 */
/*-------------------------------------------------------------*/

{ struct elist *p_x,*p_s,*p_d,*p_interm,*p,*memp;
  int k; /* indice dans ss_g */

gph_sts();
p_x = p_res;
k = 0;
nbr_ss_g = 0;

while (p_x != NULL)

  { p_s = (struct elist *) malloc(sizeof(*p_s));
    p_s->ident = p_x->ident;
    p_s->type = p_x->type;
    p_s->next = NULL;

    p_d = (struct elist *) malloc(sizeof(*p_d));

    gph_adj(p_s);
    p_interm = p_res;
    lis_unio(p_s,p_interm);
    p_d = p_res;


 while (lis_egali(p_d,p_s) == 0)

      { p_s = p_d;
        gph_adj(p_s);
        p_d = p_res;

      }

    /* p_d contient la liste des sommets du sous-graphe */

    /* recopiage dans ss_g[k]                           */

    lis_tri(p_d);
    p = p_res;

    ss_g[k] = (struct elist *) malloc(sizeof(*ss_g[k]));
    p_der = ss_g[k];
    nbr_ss_g++;

    while (p != NULL)

      { memp = p_der;
        lis_ajout(p->ident,p->type);
```

```
            p = p->next;
          }

        memp->next = NULL;
        k++;
        lis_diff(p_x,p_d);
        p_x = p_res;

      }

} /* end gph_ss_g */

/*------------------------------------------------------------------ */
```

```
gph_lg_ch(id_r,id_s,ss_gph)

/*------------------------------------------------------------ */
/* cette fonction renvoie la longueur d'un chemin minimum     */
/* entre le sommet id_r (de type R) et le sommet id_s (de     */
/* type S).                                                   */
/*                                                            */
/* les deux sommets appartiennent au meme sous-graphe         */
/* ss_gph                                                     */
/*------------------------------------------------------------ */

int id_r,id_s;
struct elist *ss_gph;

{struct elist *p_s,*p_d,*p_interm_1,*p_interm_2;
 int k;


 p_s = (struct elist *) malloc(sizeof(*p_s));
 p_s->ident = id_r;
 p_s->type = 'R';
 p_s->next = NULL;
 p_d = (struct elist *) malloc(sizeof(*p_d));
 p_d->ident = id_r;
 p_d->type = 'R';
 p_d->next = NULL;
 k = 0;

 while (lis_appart(p_s,id_s,'S') == 0)

    { lis_diff(ss_gph,p_d);
      p_interm_1 = p_res;
      gph_adj(p_s);
      p_interm_2 = p_res;
      lis_inter(p_interm_1,p_interm_2);
      p_s = p_res;
      lis_unio(p_d,p_s);
      p_d = p_res;


      k++;
    }

 return(k);

}  /* end gph_lg_ch */

/*------------------------------------------------------------ */
```

```
gph_sts()

/*-------------------------------------------------------------- */
/*  cette fonction renvoie en p_res la liste de tous les       */
/* sommets du graphe                                           */
/*-------------------------------------------------------------- */

{ struct elist *memp;
  int i,j;

p_res = (struct elist *) malloc(sizeof(*p_res));
p_der = p_res;
memp = p_res;

for (i = 0;i <= nbr_spt_sel;i++)

    {memp = p_der;
     lis_ajout(i,'S');
    }

for (j = 0;j < NBR_MED;j++)

    if (CONNEC(DIM_RC+1,j) == 1)

      {memp = p_der;
       lis_ajout(j,'R');
      }

memp->next = NULL;

} /* end gph_sts */

/*-------------------------------------------------------------- */
```

```
gph_simpl()

/*------------------------------------------------------------ */
/* cette fonction simplifie le graphe en mettant a 0 les      */
/* colonnes qui ne contiennent qu'un seul 1                   */
/*------------------------------------------------------------ */

{ int cpt;
  int i,j,k;

k = 0;

for (j = 0;j <= NBR_MED;j++)

   {cpt = 0;
    for (i = 0;i <= DIM_RC;i++)

      {  if (CONNEC(i,j) == 1)
  {k = i;
   cpt++;
         }
      }

      if (cpt == 1)
      CONNEC(k,j) = 0;
      if (cpt > 1)
      CONNEC(DIM_RC+1,j) =1;
    }
} /* end gph_simpl */

/*------------------------------------------------------------ */
```

```
gph_spt_isol()

/*------------------------------------------------------------------ */
/* cette fonction renvoie dans isol_s les symptomes isoles          */
/* du graphe, c'est-a-dire les symptomes qui n'ont aucun            */
/* remede commun avec aucun autre symptome du graphe                */
/*                                                                  */
/* nbr_spt_isol contient le nombre de symptomes isoles              */
/*------------------------------------------------------------------ */

{int i,j,cpt;
 struct elist *memp;

nbr_spt_isol = 0;
isol_s = (struct elist *) malloc(sizeof(*isol_s));
p_der = isol_s;
memp = p_der;
i = 0;

while (i < nbr_spt_sel)

  {cpt = 0;
   for (j = 0;j < NBR_MED;j++)
   cpt = cpt + CONNEC(i,j);
   if (cpt == 0)
     {nbr_spt_isol++;
      memp = p_der;
      lis_ajout(i,'S');
     }
   i++;
  }
  memp->next = NULL;

} /* end gph_spt_isol */

/*------------------------------------------------------------------ */
```

```
gph_adj(p_lis)

/*-------------------------------------------------------------- */
/* cette fonction renvoie la liste des scmmets adjacents       */
/* aux sommets de la liste p_lis                               */
/*                                                             */
/* rem : si cpt = 0 en fin d'adj alors p_lis contient un       */
/* seul sommet S isole                                         */
/*-------------------------------------------------------------- */

struct elist *p_lis;

{int cpt,fin;
 int i,j;
 struct elist *memp;

gph_an_li(DIM_RC+2);
gph_an_co(NBR_MED+1);

p_res = (struct elist *) malloc(sizeof(*p_res));
p_der = p_res;
memp = p_res;
fin = 0;
cpt = 0;

while (fin == 0)

   {if (p_lis->type == 'R')
    for (i = 0;i<=DIM_RC;i++)

     {  if (CONNEC(i,p_lis->ident) == 1 &&
            CONNEC(i,NBR_MED+1) == 0)

   {CONNEC(i,NBR_MED+1) = 1;
    memp = p_der;
    lis_ajout(i,'S');
    cpt++;
          }
     }
    else for (j = 0;j<=NBR_MED;j++)

   {  if (CONNEC(p_lis->ident,j) == 1 &&
          CONNEC(DIM_RC+2,j) == 0)

        {CONNEC(DIM_RC+2,j) = 1;
  memp = p_der;
  lis_ajout(j,'R');
  cpt++;
                }
          }
    if (p_lis->next == NULL)
    fin = 1;
    else p_lis = p_lis->next;
```

```
   }
memp->next = NULL;
if (cpt == 0) p_res = NULL;

}
/*------------------------------------------------------------------ */
```

```
gph_an_co(j)

/*------------------------------------------------------------------ */
/* fonction de mise a 0 de la colonne j                              */
/*------------------------------------------------------------------ */

int j;

{ int i;


for (i = 0;i <= DIM_RC;i++)
   CONNEC(i,j) = 0;

}

/*------------------------------------------------------------------ */



gph_an_li(i)

/*------------------------------------------------------------------ */
/* fonction de mise a 0 de la ligne i                                */
/*------------------------------------------------------------------ */

int i;

{ int j;


for (j = 0;j <= NBR_MED;j++)
   CONNEC(i,j) = 0;

}


/*------------------------------------------------------------------ */
```

```
lis_inter(p_1,p_2)

/*------------------------------------------------------------- */
/*  fonction d'intersection de deux listes dans une            */
/* troisieme                                                   */
/*                                                             */
/* Rem.  : l'une ou l'autre des liste peut etre vide           */
/* les deux listes peuvent etre vides                          */
/*                                                             */
/* p_der permet d'ajouter un element a la liste resultat       */
/* la liste resultat se trouve en p_res                        */
/*------------------------------------------------------------- */

struct elist *p_1,*p_2;

{int cpt,fin;
 struct elist *memp;

 cpt = 0;
 fin = 0;
 p_res = (struct elist *) malloc(sizeof(*p_res));
 p_der = p_res;
 memp = p_res;
 if ((p_1 == NULL) || (p_2 == NULL))
 fin = 1;

 while (fin != 1)

   { if (lis_appart(p_2,p_1->ident,p_1->type) == 1)
     {memp = p_der;
      lis_ajout(p_1->ident,p_1->type);
      cpt++;}

     if (p_1->next == NULL)
     fin = 1;
     else p_1 = p_1->next;

   }

memp->next = NULL;
if (cpt == 0)
p_res = NULL;

} /* end intersection */

/*------------------------------------------------------------- */
```

```
lis_unio(p_1,p_2)

/*------------------------------------------------------------ */
/*  fonction d'union de deux listes dans une troisieme        */
/*                                                            */
/* Rem.  : l'une ou l'autre liste peut etre vide              */
/* les deux listes peuvent etre vides                         */
/*                                                            */
/* p_der permet d'ajouter un element a la liste resultat      */
/* la liste resultat se trouve en p_res                       */
/*------------------------------------------------------------ */

struct elist *p_1,*p_2;

{int cpt,fin;
 struct elist *memp,*me_p_1;

 cpt = 0;
 fin = 0;
 me_p_1 = p_1;
 p_res = (struct elist *) malloc(sizeof(*p_res));
 p_der = p_res;
 memp = p_res;
 if (p_1 == NULL)
 fin = 1;
 while (fin == 0)

   {memp = p_der;
    lis_ajout(p_1->ident,p_1->type);
    cpt++;
    if (p_1->next == NULL)
    fin = 1;
    else p_1 = p_1->next;
   }
fin = 0;
if (p_2 == NULL)
fin =1;
while (fin == 0)

  {if (lis_appart(me_p_1,p_2->ident,p_2->type) == 0)
   {memp = p_der;
    lis_ajout(p_2->ident,p_2->type);
    cpt++;}
   if (p_2->next == NULL)
   fin = 1;
   else p_2 = p_2->next;
  }
memp->next = NULL;
if (cpt == 0)
p_res = NULL;

} /* end lis_unio */

/*------------------------------------------------------------ */
```

```
lis_diff(p_1,p_2)

/*------------------------------------------------------------ */
/* fonction de difference de deux listes dans une             */
/* troisieme                                                  */
/*                                                            */
/* Rem.  : l'une ou l'autre liste peut etre vide              */
/* les deux listes peuvent etre vides                         */
/*                                                            */
/*                                                            */
/* un element est ajoute a la liste resultat si il            */
/* appartient a la premiere liste et pas a la deuxieme        */
/*                                                            */
/* p_der permet d'ajouter un element a la liste resultat      */
/* la liste resultat se trouve en p_res                       */
/*------------------------------------------------------------ */

struct elist *p_1,*p_2;

{int cpt,fin;
 struct elist *memp;

 cpt = 0;
 fin = 0;
 p_res = (struct elist *) malloc(sizeof(*p_res));
 p_der = p_res;
 memp = p_res;
 if (p_1 == NULL)
 fin = 1;

 while (fin == 0)

   {if (lis_appart(p_2,p_1->ident,p_1->type) == 0)
    {memp = p_der;
     lis_ajout(p_1->ident,p_1->type);
     cpt++;}
    if (p_1->next == NULL)
    fin = 1;
    else p_1 = p_1->next;
   }

memp->next = NULL;
if (cpt == 0)
p_res = NULL;

} /* end lis_diff */

/*------------------------------------------------------------ */
```

```
lis_egali(p_1,p_2)

/*------------------------------------------------------------ */
/* fonction de test d'égalité de deux listes                   */
/*                                                             */
/* Rem.  : l'une ou l'autre liste peut etre vide               */
/* les deux listes peuvent etre vides                          */
/*                                                             */
/* cette fonction renvoie                                      */
/*               1 si egalite entre les deux listes            */
/*               0 si inegalite entre les deux listes          */
/*------------------------------------------------------------ */

struct elist *p_1,*p_2;

{struct elist *me_p_1;
 int fin;

me_p_1 = p_1;
if (p_1 == NULL)
  if (p_2 == NULL)
    return(1);
  else return(0);
else if (p_2 == NULL)
       return(0);

fin = 0;
while (fin == 0)

  {if (lis_appart(p_2,p_1->ident,p_1->type) ==0)
   return(0);
   if (p_1->next == NULL)
   fin = 1;
   else p_1 = p_1->next;
  }

fin = 0;
while (fin == 0)

  {if(lis_appart(me_p_1,p_2->ident,p_2->type) == 0)
   return(0);
   if (p_2->next == NULL)
   fin =1;
   else p_2 = p_2->next;
  }

return(1);

} /* end lis_egali */

/*------------------------------------------------------------ */
```

```
lis_ajout(ident_el,type_el)

/*------------------------------------------------------------ */
/* fonction d'ajout d'un element en fin de liste              */
/*                                                            */
/* on ajoute en fin de liste, en p_der                        */
/*                                                            */
/* p_der : endroit ou ajouter MAJ par cette fonction          */
/*         la fonction renvoie une nouvelle place où          */
/*         ajouter                                            */
/*------------------------------------------------------------ */

int ident_el;
unsigned char type_el;

{ p_der->ident = ident_el;
  p_der->type  = type_el;
  p_der->next = (struct elist *)
                malloc(sizeof(*p_der->next));
  p_der = p_der->next;

 } /* end lis_ajout */

/*------------------------------------------------------------ */
```

```
lis_appart(p_lis,ident,type)

/*------------------------------------------------------------- */
/* fonction de verification d'appartenance d'un element a      */
/* une liste                                                   */
/*                                                             */
/* Rem.  : la liste peut etre vide                             */
/*                                                             */
/* retourne 1 si l'element appartient a la liste               */
/*          0 sinon                                            */
/*------------------------------------------------------------- */

struct elist *p_lis;
int ident;
unsigned char type;

{ int present,fin;

  fin = 0;
  present = 0;
  while (fin == 0)

    { if (p_lis != NULL)
      if((p_lis->ident == ident) && (p_lis->type == type))
       {fin = 1;
        present = 1;}
        else p_lis = p_lis->next;
      else fin = 1;
      }
   return(present);

} /* end lis_appart */

/*------------------------------------------------------------- */
```

```
lis_tri(p_lis)

/*-------------------------------------------------------------- */
/* cette fonction trie une liste en mettant en tete de        */
/* liste les elements de type S classe en ordre croissant     */
/*                                                            */
/* la liste contient au moins un element de type S            */
/*                                                            */
/* la liste resultat se trouve en p_res                       */
/*-------------------------------------------------------------- */

struct elist *p_lis;

{ struct elist *p_tri,*p_diff,*p_interm,*p_1,*p_mem,*p_mem_2,
                *memp;
  int min,fin;

  p_1 = p_lis;
  p_tri = (struct elist *) malloc(sizeof(*p_tri));
  p_mem_2 = p_tri;
  fin = 0;

  while (fin == 0)

  {
   p_mem = p_1;
   min = 50;
   fin = 1;
   while (p_1 != NULL)

    {
      if (p_1->type == 'S')
      {fin = 0;
       if (p_1->ident < min)
       min = p_1->ident;
      }
     p_1 = p_1->next;

    }

   if (fin == 0)

     {p_diff = (struct elist *) malloc(sizeof(*p_diff));
      p_diff->ident = min;
      p_diff->type = 'S';
      p_diff->next = NULL;
      lis_diff(p_mem,p_diff);
      p_1 = p_res;
      p_der = p_mem_2;
      memp = p_der;
      lis_ajout(min,'S');
      p_mem_2 = p_der;
     }
  }
```

```
    memp->next = NULL;
    lis_unio(p_tri,p_mem);
    p_tri = p_res;

}  /* end lis_tri */

/*------------------------------------------------------------------ */
```

```
/*************************************************************/
/*              PROGRAMMES AIDANT AUX TESTS                  */
/*************************************************************/


aff_mat()

/* affichage d'une partie de la matrice CONNEC */

{int k,l;

printf("\n\n matrice CONNEC \n\n");

for (k = 0;k < DIM_RC;k++)

  {for (l = 0;l < NBR_MED;l++)
   if (CONNEC(k,l) == 1)
   printf("CONNEC(%d,%d) = 1\n",k,l);
  }

} /* end aff_mat */

/*-------------------------------------------------------- */




aff_lis(p)

/* affichage de la liste p                                  */

struct elist *p;

{ int fin;

  fin = 0;
  if (p == NULL)
  {printf("liste vide\n\n");
   fin = 1;}

  while (fin != 1)
  {printf("%d  %c\n",p->ident,p->type);
   if (p->next == NULL)
   fin = 1;
   else p = p->next;
  }
  printf("\n");
}

/*-------------------------------------------------------- */
```